BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images may include (but are not limited to):

- BLACK BORDERS
- TEXT CUT OFF AT TOP, BOTTOM OR SIDES
- FADED TEXT
- ILLEGIBLE TEXT
- SKEWED/SLANTED IMAGES
- COLORED PHOTOS
- BLACK OR VERY BLACK AND WHITE DARK PHOTOS
- GRAY SCALE DOCUMENTS

IMAGES ARE BEST AVAILABLE COPY.

As rescanning documents *will not* correct images, please do not report the images to the Image Problem Mailbox.

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
25 January 2001 (25.01.2001)

PCT

(10) International Publication Number
WO 01/06725 A2

(51) International Patent Classification[7]: H04L 29/00

(21) International Application Number: PCT/IL00/00392

(22) International Filing Date: 4 July 2000 (04.07.2000)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
130796 5 July 1999 (05.07.1999) IL
132628 28 October 1999 (28.10.1999) IL

(71) Applicants *(for all designated States except US)*: **CORESMA LTD.** [IL/IL]; Ha'melacha Street 16, 48091 Rosh Ha'ayin (IL). **BRIGHTCOM TECHNOLOGIES LTD.** [IL/IL]; Ha'melacha Street 16, 48091 Rosh Ha'ayin (IL).

(72) Inventors; and

(75) Inventors/Applicants *(for US only)*: **LIFSHITZ, Joseph** [IL/IL]; Kibush Ha'avoda Street 12, 46322 Herzlia (IL). **KAHN, Ran** [IL/IL]; Harei Yehuda 50, 55900 Ganei Tikvah (IL). **COHEN, Shlomo** [IL/IL]; Sderot Golda Meir 20, 42345 Netanya (IL).

(74) Agents: LUZZATTO, Kfir et al.; Luzzatto & Luzzatto, P.O. Box 5352, 84152 Beer-Sheva (IL).

(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CR, CU, CZ, DE, DK, DM, DZ, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, US, UZ, VN, YU, ZA, ZW.

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published:
— *Without international search report and to be republished upon receipt of that report.*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

WO 01/06725 A2

(54) Title: PROCESS AND SYSTEM FOR CARRYING OUT PARALLEL PACKET PROCESSING

(57) Abstract: A process for carrying out the processing of received and transmitted data streams made of packets in a Packet Processor, the process comprising: (A) Receiving into a pipeline a stream of packets, each packet mainly comprises a header and a payload section; analyzing the header of the packet for validity, timing information and addressing information; simultaneously with the analysis of the header, conducting a validity check of the payload; optionally, if the payload is encrypted, carrying out decryption of the payload. If any time during the above steps, at least one of the check fails; terminating the processing of the packet. (B) Providing a transmitting pipeline. Receiving by the pipeline a payload from a Host, and at least a destination address. Creating an associated header and adding validity information to said header; associating the header with the payload and forwarding in the pipeline; optionally, encrypting the payload; any time during the performance of the above, if an error occurs, terminating the processing of said packet for transmission.

PROCESS AND SYSTEM FOR CARRYING OUT PARALLEL PACKET PROCESSING

Field of the Invention

The invention generally relates to broadband communication systems for conveying digital data. More particularly, the invention relates to a system and process for carrying out parallel packet processing, particularly in broadband modems.

Background of the Invention

The recent development of the Internet, and the development of computer systems directed towards working in wide and global networks have significantly increased the use of modems. Modems for broad band communication are now required to deal with a very high rate of data transfer in a variety of environments.

In the previous decade, modems were mostly used for transferring digital data over a telephone line. In today's communication systems, complex modems are also used for conveying data over other types of mediums, for example, over TV cables, or satellite links. Wireless and/or broad band modems are used, for example, in mobile communications, e.g., for communicating with cells or satellites, etc. Routers, which are widely used in networks, are also an example for such application.

The rapid development of modems for these and other purposes, and the fast and frequent increase in the data transfer rate have increased the necessity to frequently develop and define new standards and protocols for communications. The frequent introduction of new standards, and the increase in the transfer rate have generally required the replacement of older standards, by those compatible with the new standards, as the older modems could not comply with the newer standards, could not adapt to the change in the data transfer rate, or could not be reconfigured.

Generally, any modem comprises two main sections. The first section, the modulator-demodulator section, is a mixed signal section for interfacing between the modem and the medium of transfer, for example, a telephone line, TV cable, or the air, in the case of a wireless modem (hereinafter, when the term "modulator-demodulator" is used, it should be understood to refer to the above-indicated section of the whole apparatus called modem. Moreover, the said two sections, the modulator and demodulator sections should be viewed in their broader term, like a transmitter and receiver accordingly, as they also include logic means for managing the transmission and reception. When the term "modem" is used, it should be understood to refer to the whole apparatus commonly called modem).

The second section of any modem is digital, generally referred to as the Media Access Control (MAC) module. The MAC module operates in the Media Access Control layer. The purpose of the MAC module is to manage and handle the transfer of digital data between the modulator-demodulator section of the modem and a host, in which the higher level layers are implemented, is generally located external to the modem casing, and *vice versa*.

The MAC module, is the heart of any modem. The MAC module receives a sequence of data stream from a host, creates packets of data that are then transmitted by the modulator, or receives such packets of data from the demodulator and creates a data sequence from them. Of course, these packets also contain additional information that the two communicating modems may need for assuring a reliable communication, i.e., for enabling the recovery of the data at the receiving end. More particularly, the Media Access Control handles error correction, regulates the data flow, handles the handshaking between two network peers, and optionally encrypts or decrypts the transferred data, when necessary, etc. Other functions of the MAC module, when used e.g., in a modem for TV cables are, to carry out the synchronization with the CMTS (Cable Modem Termination System), to manage upstream transmission allocation mechanism, to operate transmission of data on time slot boundaries, and to filter the received headers from the received data. Of course, the MAC module should comply

with certain predefined standards, in order to enable the modem to properly communicate both with other modems, and with the host.

Of course, it is essential for the MAC module to handle its tasks in a fast and reliable manner, as the performance of the whole modem greatly depends on the performance of this module. In the existing modems, particularly those working at a very high rate, for example, modems for conveying digital data over TV cables, or those for communicating over fiber-optic links, this is not an easy task, as the amount and rate of the data that the MAC module has to handle are very high.

Efforts have been made to use a processing unit for carrying out many of the tasks of the MAC module, however, with limited success. High speed packet processing poses serious challenges to a single general purpose processor. This is the main reason why many existing modems use hard-wired logic for some of the lower level tasks of the MAC layer, while a high speed processor, if such exists, takes control only at the packet level or IP (Internet Protocol) level. More particularly, in the existing modems the data is processed by a plurality of gates for carrying out the MAC and packet handling tasks. This configuration is rigid, and cannot be changed or reconfigured when a necessity arises.

Copending Israeli Application No. 130796, filed July 5, 1999, assigned to a co-assignee hereof, discloses a structure for a Packet Processor for communication applications, particularly for modems. The Packet Processor of IL 130796 provides a tool for handling a high rate of data and performing the above-mentioned tasks in an efficient manner. A particular object of the invention of Copending Israeli Application No. 130796 is the providing of a Packet Processor for the broadband and wide band communication schemes, particularly for implementing functions that must be handled in real time. In one particular case, the Packet Processor of Copending Israeli Application No. 130796 is used as a MAC module of a modem.

The invention of copending IL 130796 provides a structure for said Packet Processor, which can easily adopt new communication standards, and change of data rate, when necessary.

The invention of copending IL 130796 further provides a structure for a Packet Processor that can communicate with different types of peripherals. In a particular case when the Packet Processor of copending IL 130796 is used as a MAC module for modem, the packet processor is able to communicate with different types of PHY (modulator-demodulator) chips. The said Packet processor can be easily integrated in a single Very Large Scale Integration (VLSI) chip.

Furthermore, the Packet Processor of IL 130796 is general-purpose, and can be used in other communications applications, for various purposes, due to its programming characteristics.

The said Packet Processor of Copending IL 130796 is characterized by that it comprises mainly of two separate parts, a first part for handling a flow of received data (hereinafter, this will be referred to as the "receiving part"), and a second part (hereinafter, the "transmitting part") for handling the flow of the transmitted data. Each of the said two parts includes at least one processing-unit, and preferably at least two processing-units. Data from a host processor is provided to the Packet Processor and conveyed over a transmitting Tubular Bus, while processed by the one or more processing-unit/s of the transmitting part. These one or more processing-units of the transmitting part frame the data, and combine with it some additional information, such as additional bits for enabling data error detection and correction (e.g., CRC - Cyclic Data Redundancy). The said processing-unit/s of the transmitting part further organize the data in packets that are then optionally encrypted, conveyed over the Tubular Bus, at an appropriate timing slots, to the modulator for transmission. Digital data that is received from the demodulator is conveyed over the receiving Tubular Bus, while being processed by the receiving one or more processing-unit/s. The receiving processing-units deframe the received data, detect and correct errors by processing the error correction data associated

with the received data, and decrypt the data, when necessary. The deframed data is then conveyed to its destination, a host processor. The Packet Processor in its basic structure further comprises a management bus by which the receiving and the transmitting processing-units, communicate with other components of the Packet Processor, or with external components.

It is one object of the present invention to provide a process and system for processing a transmitted data stream and a received data stream in a packet processor that comprises two separate parts, a receiving part and a transmitting part, each of said parts comprises at least one processing unit. The received and transmitted data streams are processed in parallel, while flowing over the received and transmitted tubular buses respectively. The method and system of the invention may be carried out on the packet processor as described in IL 130796, or on others packet processors having a different structure. The process of IL 130796 is referred to herein for convenience of explanation, but the skilled person will easily appreciate different packet processors in which the present invention can be exploited.

It is still another object of the invention to make the method and system of the invention being applicable for various of the existing data protocols for broadband modems, such as MCNS, DOCSIS and DVB/DAVIC, possibly with minor modifications.

It is still an object of the invention to make this method capable of processing the received and transmitted data streams in a fast and efficient manner.

Other objects and purposes of the invention will become apparent as the description proceeds.

Summary of the Invention

The invention relates to a process for carrying out the processing of received and transmitted packet streams the process comprising:

a. Providing two sections, a transmitting section and a receiving section, each section comprises of at least one processing unit;

b. In the receiving section:

b.1. Receiving a data stream of packets from a demodulator section, passing the stream through a receiving pipeline, while processing the data by said at least one processing unit of the receiving part, processing tasks are performed simultaneously when possible;

b.2. When all processing tasks are completed, conveying the processed packets to a Host;

c. In the transmitting section:

c.1. Receiving a packets stream from a Host section, passing the stream through a transmitting pipeline, while processing the packet

stream by said at least one processing unit of the transmitting section, processing tasks over the data stream are performed simultaneously when possible;

c.2. When all processing tasks are completed, conveying the processed packets into a modulator for transmission;

Preferably, the transmitting pipeline and the receiving pipeline are one way pipelines comprising a plurality of segments. The said pipelines may comprise hardware segments, software segments, or a combination of hardware and software segments.

Preferably, the invention relates to a process for carrying out processing of received and transmitted data streams made of packets in a Packet Processor, the process comprising:

A. a receiving process comprising:

a. Providing a receiving pipeline, and at least one processing unit ;

b. Receiving into said pipeline a stream of packets from a demodulator section, each packet mainly comprises a header and a payload section;

c. Extracting a header from the received packet;

d. Analyzing the header of the packet for checking the header validity and for extracting from it timing information (when exists), and

addressing information, while forwarding the payload downstream the pipeline.

e. Simultaneously with the analysis of the header, conducting a validity check of the payload; if the validity check of the payload is found valid, forwarding the payload downstream the pipeline;

f. Optionally, if the payload is encrypted, carrying out decryption of the payload; when the decryption is completed, forwarding the payload towards the Host;

g. Simultaneously with one of steps d-f, comparing the addressing information as extracted in step d with a list of valid addresses;

h. Optionally, if any time during the performance of steps d-f, at least one of the following failures occurs:

- the validity check of the header (step d) fails; or
- the validity check if the payload fails (step e); or
- the addressing information in the header is found to be different from any address in said list of valid addresses (step g); or
- a packet could not be completely received;

Terminating the processing of the packet, dumping the information relating to the said packet, and initiating an error message that corresponds to the type of failure.

B. A transmitting process comprising:

a. Providing a transmitting pipeline, and at least one processing unit;

b. Receiving by the pipeline a payload from a host, and at least a destination address; forwarding the same in the pipeline;

c. Framing the payload, including at least creating an associated header, and simultaneously with said header creating, adding validity information to said header payload; associating the header with the payload and forwarding in the pipeline;

d. Optionally, encrypting the payload;

e. Simultaneously with steps b-d, preparing a proper transmission timing slot for the transmission of the packet;

f. Any time during the performance of steps b-e, if an error occurs, terminating the processing of said packet for transmission, dumping the information associated with said packet, and initiating an error message notifying the type of the error;

g. If no error is detected in step f, forwarding the resulting packet to a modulator for transmission.

According to one embodiment of the invention the Packet Processor for carrying out the above process comprises:

A. A receiving section comprising:

a. A receiving tubular bus;

b. At least one processing unit connecting between segments of the tubular bus;

c. One FIFO before and one FIFO after any processing unit on the tubular bus;

d. Optionally at least one address filter for comparing an address extracted from a header of a received packet with a list of valid addresses;

B. A transmitting section comprising:

a. A transmitting tubular bus;

b. At least one processing unit connected between segments of the tubular bus;

c. One FIFO before and one FIFO after any processing unit on the transmitting tubular bus;

C. A PHY interface for receiving data from a modulator/demodulator unit and conveying it to the receiving tubular bus, and for receiving data from the transmitting bus and conveying it to the said modulator/demodulator unit for transmission;

D. A Host interface for receiving data from a Host and conveying it to the transmitting tubular bus, and for receiving data from the receiving tubular bus and for conveying it to the Host;

E. At least one bus for conveying management information between different components of the Packet Processor; and

F. Timing and control means for administering the operation of the Packet Processor, and particularly for providing the allocation of transmission slots for the transmitting of packets.

The invention also relates to a system for carrying out the processing of received and transmitted packet streams, the system comprising:

a. a modulator/demodulator section for transmitting and receiving packet streams respectively;

b. A Packet Processor comprising a transmitting section for preparing packets for transmission and a receiving section for extracting the data from a received packet stream, each of said transmitting and receiving sections comprising at least one processing unit, a tubular bus connecting the processing units, and a FIFO before and a FIFO after each processing unit on the tubular bus;

c. A Host for receiving processed data from the receiving section of the Packet Processor, and for conveying data for transmission into the transmitting section of the Packet Processor; and

d. A process according to claim 1 for processing received and transmitted packet streams.

Preferably, the process of the invention is carried out by a microcode residing in the modulator/demodulator section for carrying out real-time processing in the Packet Processor. In that case, the microcode part of the process preferably comprises:

I. A receiving section comprising:

a. A one way pipeline made of segments, said pipeline receives data streams from the modulator/demodulator section;

b. MacroInstructions embedded with the data flowing over said pipeline, for conveying commands between different stages that are connected to the pipeline;

c. A Byte Stream Reception stage for receiving framed packets from the pipeline, and extracting PMD frames from the data stream and forwarding the resulting frames into a De-Framing Stage via the pipeline;

d. A De-Framing stage for receiving data from the Byte Stream Reception Stage via the pipeline, extracting a MAC Frame, and dumping PMD header and validity data; forwarding the resulting stream into an Initial Packet Analysis Stage;

e. An Initial Packet Analysis stage for receiving data from the De-Framing stage via the pipeline, analyzing the said MAC header, forwarding timing information into a Timing Analysis stage, carrying out a validity check on the MAC header, and forwarding the resulting packet stream into the pipeline;

f. A Packet Filtering stage for comparing addressing information that is extracted from the MAC header with a list of valid addresses; if the result of the comparison is negative, dumping the current packet; if the result of the comparison is positive, forwarding the stream into an Advanced Packet Analysis stage;

g. An Advanced Packet Analysis stage for carrying out validity check on the payload, forwarding transmission timing information into a TX Information Filter, and forwarding the resulting payload into a Flow Transmission to Host stage, or optionally into a Security stage; and

h. A Flow Transmission to Host stage for receiving a payload, and conveying same into a Host;

i. An optional Security stage for decrypting the payload, if the payload is encrypted.

II. A transmitting section comprising:

a. A one way transmitting pipeline made of segments, said pipeline receives unframed payloads packet information from the Host;

b. A Flow Reception From the Host stage for receiving unframed payload packets information from Host via the transmitting pipeline, and forwarding the same into a Framing stage via the transmitting pipeline;

c. A Framing stage for constructing a MAC header, combining validity information with the header and payload, and forwarding the resulting framed data packets into a Byte Stream Transmission stage or optionally a Security stage via the transmitting pipeline;

d. A Security stage for optionally receiving framed data packets, encrypting same, and forwarding the same into a Byte Stream Transmission stage;

e. A Byte Stream Transmission stage for conveying the framed, and optionally encrypted data packets into a modulator/demodulator section for transmission; and

f. A Timer and Transmission Management stage for assigning transmission timing slots for framed and optionally encrypted data that is conveyed into the modulator/demodulator section.

Preferably, the Transmission Management stage assigns transmission timing slots based on information collected from the analysis of MAC frames by the Advanced Packet Analysis stage of the receiving section, and on information received from the Flow Reception From Host stage of the transmission section. Preferably, the information for assigning transmission timing slots is stored in a memory that is shared by the transmitting section and the receiving section.

Preferably, the system of the invention further comprises a CMMS 867 high level language residing in the host, and a driver for interfacing between the CMMS 867 and the microcode residing at the MAC processor for carrying out the process of the invention. Preferably, the system further comprises a GUI high-level language code for enabling a user of the process to monitor and configure the process.

Preferably, the validity check is at least a checksum check. More preferably, the validity check is a checksum check and error correction.

Brief Description of the Drawings

In the drawings:

- Fig. 1 is a scheme showing the input/output buses of the packet processor of IL 130796;
- Fig. 2 is a more expanded scheme of Fig. 1, showing the input/output buses of the packet processor, together with some components in the periphery of the packet processor of IL 130796;
- Fig. 3 shows a basic structure of a packet processor, according to one embodiment of IL 130796;
- Fig. 4 shows a structure of a packet processor, according to another embodiment of IL 130796;
- Fig. 5 shows a structure of one processing unit in the packet processor of Fig. 4, according to an embodiment of IL 130796;
- Fig. 6 shows a structure of a packet processor, according to still another embodiment of IL 130796;
- Fig. 7 shows a structure of one processing unit in the packet processor of Fig. 6, according to IL 130796;
- Fig. 8 is a general block diagram illustrating the process of the present invention; and

- Fig. 9 is a more detailed block diagram illustrating the process of the present invention.

Detailed Description of Preferred Embodiments

Copending application IL 130796, discloses a Packet Processor that comprises a plurality of processing units, in which tasks are distributed in an efficient way, thereby providing a Packet Processor that can function in a high rate environment, and which is programmable, thereby providing flexibility and adaptation to different types of environments, standards, protocols and purposes. The general structure of the Packet Processor of IL 130796 is suitable, probably with a few necessary modifications, for many communications applications, in modems, routers or others.

The specific examples given in IL 130796 are particularly suitable for use in broad band modems, for example, modems for TV cables. Such modems are now working mostly according to the following standards: MCNS (Multimedia Cable Network System)/DOCSIS (Data Over Cable System Interface Specifications), DVB (Digital Video Broadcast), P/DAVIC (Digital Audio Video Council), and IEEE802.14 (IEEE's Cable TV MAC and physical Protocol Working Group).

Figure 1 illustrates a basic structure of the Packet Processor of IL 130796, according to one embodiment of the invention. The Packet Processor 1 is a

Field Programmable Processors Array (FPPA) having four main interfaces for communicating with other devices and components outside the Packet Processor.

The processor array 2 includes plurality of processors, their associated coprocessors, and buses. The structure of the processor array is elaborated in more detail hereinafter.

A host interface 3 connects the Packet Processor to a host. The host is generally a computer, a CPU, or another processing or calculating device which, in addition to the performance of other functions, is the source or the destination of a data stream that is transmitted or received by the modem.

A general-purpose interface 7 connects the Packet Processor with components such as, for example, a keyboard, a debugging processor, a tuner, flush memory, etc. The General Purpose Bus 7 is a relatively slow bus, as it is generally used for "servicing" the processor.

The External Bus Interface 4 connects the plurality of processing units of the Packet Processor with a main memory unit, or optionally with other components on the board of the modem, etc. According to a preferred embodiment of the invention, the main memory unit is external to the Packet Processor, and each processing unit comprises an internal

Instruction Cache (Icache) containing a portion of the code needed for its operation. When there is a miss in any of the said Icaches, a transfer of the missing code is transferred to the relevant internal cache via the External Bus Interface 4.

The PHY (physical) interface 6 connects the Packet Processor with the modulator-demodulator section of the modem. Through the PHY interface 6 a data stream is conveyed from the Packet Processor to the modulator for transmission, and a data stream is received at the Packet Processor from the demodulator. The PHY interface 6 is generally a programmable state machine, which is implemented, e.g., by RAM cells.

Fig. 2 depicts in more detail the environment of a Packet Processor according to IL 130796, and its connections with external components or devices. Numeral 2 indicates the Packet Processor. The PHY bus 16 connects the Packet Processor with the modulator-demodulator section 10 of the modem, generally a transceiver, which operates in the physical layer. The External bus 14 connects the Packet Processor 2 with the external memory unit 19, and optionally with additional external components, indicated as numeral 20. The Packet Processor 2 communicates with the host 21 via the host interface 3 and the PCI bus 13.

Hereinafter, if not otherwise specifically indicated, the term "received data" refers to the data that is conveyed to the Packet Processor 2 from the modulator-demodulator unit of the modem. The modulator-demodulator 10 receives modulated data from the medium of transfer, demodulates it, and transfers it in a form of a bit stream to the Packet Processor. In the particular case when the Packet Processor is used as a MAC module of a TV cable modem, the modulator-demodulator 10 receives the modulated data from the cable network. In that particular case the data is typically originated from a CTMS (Cable Modem Termination Modem also known as a Head End). The term "transmitted data" refers herein to the data that is conveyed to the Packet Processor 2 from a host, and, after being processed by the Packet Processor 2, the processed data is transferred to the modulator-demodulator section 10 of the modem, which in turn modulates it with a carrier, and transmits the modulated data to another modem over a medium of transfer. In the particular case when the Packet Processor 2 is used as a MAC module of a TV cable modem, the modulated data is transmitted to the cable network, where it typically reaches a CMTS (also known as a Head End).

A basic structure of the Packet Processor according to one embodiment of the invention of IL 130796 is shown in Fig. 3. The Packet Processor 1 comprises mainly two separate parts, a first part 23 for handling the flow of received data (hereinafter, this will be referred to as the "receiving part"),

and a second part 24 (hereinafter, the "transmitting part") for handling the flow of the transmitted data. Each of the said two parts mainly comprises at least one processing-unit, and preferably at least two processing-units. Data from the host 26 is provided to the Packet Processor through the PCI interface 25. This data is then conveyed over the Tubular Bus 29 to the processing-unit/s of the transmitting part 24. This processing-unit frames the data, and combines with it some additional information, such as additional bits for enabling data error detection and correction (e.g., CRC - Cyclic Data Redundancy). The said processing-unit/s 22 of the transmitting part further organizes the data in packets that are then optionally encrypted, conveyed over the Tubular Bus 29 to the PHY (Physical) Interface 40, and from it, at an appropriate timing slots, to the modulator for transmission. Digital data that is received from the demodulator is conveyed through the PHY interface 40, and over the Tubular Bus 29 to the receiving processing-unit/s 21. The receiving processing-unit/s 21 deframes the received data, detects and corrects errors by processing the error correction data associated with the received data, and decrypts the data, when necessary. The deframed data is then conveyed to the PCI interface 25, and from it to its destination, the host processor 26. A second bus, the Ring Bus 44, is actually a "management" bus of the Packet Processor. Through the Ring Bus 44 the receiving and the transmitting processing-units, 23 and 24 respectively, communicate with other components in the Packet Processor, or with external components, and over

this bus data, addresses or instructions flow between the Packet Processor components. For example, via the Ring Bus 44 the code of the Packet Processor is downloaded from an external data storage, e.g., a diskette, a flush memory/EPROM, or a CD ROM to the memory unit of the modem (the main memory unit by itself is external of the Packet Processor and is not shown in Fig. 3). Furthermore, the Ring Bus 44 may optionally be used for enabling communication of the Packet Processor with a keyboard, with a debugging unit, or with other different units or devices external to the Packet Processor. Furthermore, communication may be carried out from the host processor 26, to the Ring Bus 44 via the PCI interface and the extension to the Ring Bus 46, and from the Ring Bus to any component that is linked to it. By this way the code of the Packet Processor may also be downloaded to the Packet Processor 1 from the host through the PCI bus 25. The Packet Processor 1 further comprises FIFO storage components 60, 61, 62, and 63, for temporarily storing portions of the data flow, a control and timing unit 65 for timing and synchronizing the operation of the Packet Processor, in particular the allocation of the transmit data slot, and some external interfaces 66 for communicating with components and devices external to the Packet Processor.

Fig. 4 depicts in even more detail the structure of the Packet Processor, according to one embodiment of the invention of IL 130796. As said, the receiving and the transmitting parts 21 and 22 preferably comprise at least

two processing-units each. When two or more processing-units are used in each part, the performance of the Packet Processor is significantly improved. In the embodiment of Fig. 4, the receiving part comprises a processing-unit-1 51, and a processing-unit-2 52, and the transmitting unit comprises processing-unit-3 53, and processing-unit-4, 54. The width of the Tubular Bus of the Packet Processor of Fig. 4 is 9 bits, and the width of the Backbone Bus is 32 bits.

In the embodiment of Fig. 4, the received data is preferably processed in two phases. The first phase, done by Processing-unit-1 51, includes the processing of the header CRC (handling of the 16 bit error correction) and the deframing of the data stream, while the second phase, done by Processing-unit-2 52, includes the handling of the main CRC of the content of the packet (32 bit error correction of the data section (payload) of the packet), decryption, logical analysis including determining the length and type of the packets (management, or data), possible concatenation of packets and other related activities. The transmitted data is preferably processed in two phases. The first phase, done by Processing-unit-3 53, includes the handling of the creation of the header CRC (16 bit error correction), controlling allocation, and prioritizing the transmission sequences, and other activities related to the transmit time allocation. The second phase done by processing-unit-4 54 includes the creation of the main CRC (32 bit error correction), encryption, and framing of the transmitted data.

It should be noted here that the above allocation of tasks to the specific processing-units of the Packet Processor 1 is optional, although preferable, as the Packet Processor is programmable, and the code according to the invention is downloaded to the memory from an external source (not shown). Modifications to the code can be made at any time, and such modifications may change the allocation of the tasks to specific processing units.

Preferably, each of the four processing-units in the embodiment of Fig. 4 comprises a RISC-type (Reduced Instruction Set Computer) processor. The use of a RISC processor in the processing unit is preferable, due to its fast processing rate, simple structure, small silicon area, and flexibility of operation. Each processing unit also comprises an internal memory, which in turn is divided into a Scratch PAD RAM Memory (hereinafter, "SCRAM") and an instruction cache (ICACHE) memory. The RISC processing-units of the embodiment of Fig. 4 are characterized by having a separate access to three buses: A first, input bus 91, to the RISC processing-unit is mainly used for receiving the data flow from a FIFO in the Tubular Bus. A second, output bus 92 of each RISC processing-unit is mostly used for outputting the processed data from the processing-unit to the Tubular Bus. The third bus 93 of the processing-unit is bdirectional, and is used by the RISC processing-unit to gain access to the Backbone Bus 44, in order, for example,

to communicate with internal or external peripheral components, or to load or download data and instructions from the external main memory to the internal memory of the processing unit, i.e., the SCRAM and the cache memory. The communication of the RISC processing-units with the peripheral components via the Ring Bus 44 is used, among other purposes, for controlling the process of the downloading of the code to the memory of the Packet Processor, or for debugging purposes, etc.

As shown in Fig. 4, the Packet Processor of the invention of IL 130796 also comprises several FIFO storage elements 70, 71, 72, 73, 74 and 75. The purpose of the FIFOs is to provide a sequential temporary storage for portions of the data flow, before or after being processed by the respective processing-units of the Packet Processor. The said FIFOs enable each of the processing-units to sequentially process a flow of data, and to process and carry out operations on a portion of the data flow which is conveyed to it from a FIFO, while eliminating data loss.

Each FIFO of the Packet Processor 1 is basically a RAM stage for storing a plurality of words of data, with the addition of some logic surrounding it. The structure of all the FIFOs of the Packet Processor, 70, 71, 72, 73, 74, and 75, is generally identical, but the RAM stage size may differ from one specific FIFO to another. For example, according to one implementation of the invention, the size of the RAM stage in different FIFOs ranges between

32 to 64 words of 9 bits. The FIFO sequentially receives words of data or macro-instructions from the Tubular Bus. The macro instructions are instructions that are originated by a processing unit, embedded with the data stream in the tubular bus, and used for controlling the FIFOs. The macro instructions are discussed in more detail hereinafter. The words of data are then stored in the RAM stage of the FIFO. The logic circuitry that surrounds the RAM stage handles the progression of the words in the FIFO from the FIFO input to its output, and eliminates inputting of new words of data when the FIFO is full, or outputting a data word from it when the FIFO is empty or the processing-unit which should receive it is busy. FIFO 71, the second FIFO downstream the receiving path functions in combination with an Address Filter 84. The Address Filter 82 comprises a table of addresses. When an address is detected in a header of the data stream, a simultaneous comparison is made with all entries of this table. According to the result of the comparison, a decision is made whether the next data should be forwarded for a further processing or whether the data should be ignored. An ignoring of data may occur, for example, when the destination of this data is different from the one the packet processor operates.

The Packet Processor 1 further comprises a timer 77, which administers the operation of the module by providing numerous clock and control signals to the different components of the Packet Processor. In particular, it also

controls the essential task of slot timing assignments associated with the transmit path a task which is well defined in the relevant protocol standards, and is familiar to those skilled in the art. The bus arbiter 79 administers the use of the Backbone Bus by different components at different times, according to some hierarchy rules and priority considerations. The ICU (Interrupt Central Unit) 80 administers the functioning of the interrupts in the module.

Furthermore, several of other interfaces, generally indicated herein as External Interface stage 78, are also included in the Packet Processor of the invention for communicating with other peripherals external to the module. For example, the module comprises a serial interface, preferably programmable, capable of implementing standards such as I^2C (Inter-Integrated Circuit) standard, a JTAG (Joint Test Action Group, IEEE Standard 1149.1-1990) interface, etc. Each of the said interfaces handles communications between different types of peripheral components, and components of the Packet Processor via the Backbone-Bus. The External Bus Arbiter 99 is used for arbitrating the usage of the external bus 97,which used for accessing external memory. One application of such memory is the downloading of the application code by which the module operates from, (typically stored in an external storage means such as a diskette, a PROM, RAM memory, etc.), to the memory via the External Bus 97. Said downloading of the code is carried out each time upon turning ON of the

modem, by the External Bus Arbiter 99, via the Backbone Bus 44, to the main memory unit (not shown), which, as said, is external to the Packet Processor. The internal memory of each processing unit, according to a preferred embodiment of the invention, is a dual port memory type, wherein one port of the memory unit is connected to the Backbone Bus, and the second port of the internal memory unit is connected to a processing-unit. The structure of the processing-units, and their connection with the internal memory unit, is discussed in more detail hereinafter.

The four processing-units 51, 52, 53, and 54 of the Packet Processor, have essentially the same structure according to the invention. However, in some cases, one or more may differ from the others, in view of the specific tasks that the processing units handle. Fig. 5 depicts the structure of a preferred processing-unit, according to one embodiment of the invention. The processing-unit 100 comprises two main components, a RISC processor 101, and a co-processor 102. The use of a RISC processor is preferable in this case, as it is characterized by having a reduced instruction set, that enables the processor to have a small silicon area, and to function in very short cycles. The number of the various tasks that are performed by processor 101 of the processing unit are few and repetitive, so an RISC processor can perform these tasks most efficiently, particularly in association with co-processor 102. In the embodiment of Fig. 5, an ARC (by ARC Cores Ltd.) type RISC processor, has been selected and found preferable, as it has

separate input/output buses for the data flow, and for the fetching of instructions. The internal memory 109 of the processing unit 100 comprises an instruction cache memory section 110, preferably of the SRAM type, and a Dual Port Scratch Pad RAM 111. The Scratch Pad RAM 111 is loaded with data from the main memory unit via the Backbone Bus 44. The data in the internal memory 109 is generally sufficient for the operation of the processing unit, but when necessary, data is exchanged between the internal memory and the external memory via the Backbone Bus 44, the External Bus Arbiter 99, and the External Bus 97. The processor 101 itself communicates with the internal memory 109 via the Memory Bus 113. In order to improve the overall efficiency, some of the specific tasks of the processing unit are performed by the co-processor 102, with the supervision and control of processor 101. The communication between processor 101 and the co-processor is made via the Auxiliary Bus 105, having a width of preferably 32 bits. The processing unit 100 also comprises a clock unit 115, an Interrupt Control Unit 116, and optionally, a Debug Interface Unit 117. The Interrupt Control unit 116 provides interrupts to the Interrupt Interface 118 of processor 101, when necessary. The two FIFO storage units at the input and the output of the processing unit are indicated herein as numerals 120 and 121, respectively. These FIFOs are, in the case of processing unit 52, FIFO-2 71, and FIFO-3, 72.

The RISC processor 101 has several internal registers other than the internal register files (not shown). The main registers of the RISC processor are, the RAM Access Registers 135 for temporarily storing data to/from the internal memory 109, an In FIFO Register 136, and an Out FIFO Register 137.

Fig. 6 shows the structure of a Packet Processor according to a more preferred embodiment of the invention of IL 130796. The Packet Processor of Figs 3, 4, and 5 comprises two main buses, a Tubular Bus 29, and a Backbone Bus 44. Whenever one of the processing units needs an instruction that is not in the cache memory section 110, or a memory data that is not in the Dual Port Scratch Pad RAM 111, an access made into the external memory unit, in order to load the internal memory 109 (the cache memory section 110, or the Dual Port Scratch Pad RAM 111) with said missing content. The communication with the external memory unit is carried out by any processing unit via the Backbone Bus 44, the External Bus Arbiter 99 and the External Bus 97. However, this configuration suffers from the drawback that any access to the main memory unit eliminates simultaneous use of the Backbone Bus, which, consequently, slows the operation of the module.

The Packet Processor of Fig. 6 overcomes this drawback by having three main buses, a Tubular Bus 243, a Backbone Bus 244, and an External

Access Bus 245. In this structure, simultaneous activities can be carried out in the External Access Bus, and in the Backbone Bus, eliminating the need, for example, to disable activity on the Backbone Bus 244 while a processing unit accesses the main memory.

The Tubular Bus 242 is preferably a 9-bit bus, and the Backbone Bus 244 and the External Access Bus 245 are preferably buses of 32-bits.

The Packet Processor of Fig. 6 comprises, as before, four processing units, two processing units 251, 252 in the receiving part, and two processing units 253, and 254 in the transmitting part. The Tubular Bus 243 is actually divided into two separate buses, the receiving Tubular Bus 2430 of the receiving part, and a Tubular Bus 2431 of the transmitting part. As before, through the receiving Tubular Bus 2430 the received data from the demodulator of the modem flows, while being processed, to the host, and through the transmitting Tubular Bus 2431, a transmitted data flows, while being processed, from the host to the modulator of the modem. Of course, as before, the modulator and the demodulator are parts of the modulator-demodulator unit of the modem (not indicated in this figure). The communication of the Packet Processor with the modulator-demodulator is carried out via the PHY interface, which comprises two separate portions, a receiving PHY interface 201, and a transmitting PHY interface 202. The bidirectional communication between the Tubular Bus of the Packet

Processor and a host, is made via a PCI interface 250. Alternatively, a bidirectional communication between the Tubular Bus of the Packet Processor and one of a plurality of hosts that are connected to a local network may be carried out via an Ethernet interface 351. Such communication involves the transfer of data that has to be processed by the Packet Processor and then transmitted by the modulator, or data that has been received by the Packet Processor from the demodulator, processed, and then transferred to a host. Timer 277 provides control and timing signals to different components of the module, but primarily controlling timings related to the transmit path operations. The optional Debug Interface Unit 291 enables the debugging of the module. It is shown as operating in conjunction with processing unit-1 251, however, it may work with other processing units of the module, as well. The Interrupt Central Unit 280 administers the interrupts to different components of the module, as needed.

The Packet Processor of Fig. 6 optionally further comprises a Serial Interface 278, for providing serial communication of the Packet Processor with any external serial device or component, however, it is optional.

As said, the Packet Processor of Fig. 6 comprises four processing units, 251, 252, 254, and 255. The received data is preferably processed in two phases. The first phase, done by Processing-unit-1 251, includes the processing of

the header CRC (handling of the 16 bit error correction) and the deframing of the data stream, while the second phase, done by Processing-unit-2 252, includes the handling of the main CRC of the content of the packet (32 bit error correction of the data section (payload) of the packet), decription, logical analysis including determining the length and type of the packets (management, or data), possible concatenation of packets and other related activities. The transmitted data is preferably processed in two phases. The first phase, done by Processing-unit-3 253, includes the handling of the creation of the header CRC (16 bit error correction), timing, controlling allocation, and prioritizing the transmission sequences, and other activities related to the transmit time allocation. The second phase done by processing-unit-4 254 includes the creation of the main CRC (32 bit error correction), encryption, and framing of the transmitted data. The register file 295 comprises additional external registers for optional use of any of the processing units.

As in the embodiment of Figs. 4 and 5, the four processing units of the Packet Processor of Fig. 6 have essentially the same structure. However, this is not a limitation, and each processing unit may have a specific structure, which is most suitable for the task it handles. Each of the processing units of the Packet Processor of Fig. 6 mainly comprises a processor and co-processor, and a Sub-Unit Controller (SUC). In the processing unit 251, the processor-1 and its associated co-processor, are

indicated as stage 361, and the sub-unit controller is indicated as stage 261; in the processing unit 252, the processor-2 and its associated co-processor are indicated as stage 352, and the Sub-Unit Controller is indicated as stage 262; in the processing unit 253, the processor-3 and its associated coprocessor are indicated as stage 353, and the Sub-Unit Controller is indicated as stage 263; and in the processing unit 254, the processor-4 and its associated co-processor are indicated as stage 364, and the Sub-Unit Controller is indicated as stage 264. Each processor and its associated co-processor of any of the said processing units handle a specific task, while the SUC of that unit (261, 262, 263 or 264) handles the communication of the unit with the Backbone Bus (BB) 244, and the External Bus (EB) 245. Preferably, also in the embodiment of Fig. 6, the processors are preferably RISC type processors, as such processors are capable of performing relatively simple tasks using small silicon area, in a very rapid manner.

The structure of processing unit 251 of the Packet Processor of Fig. 6, according to IL 130796 is shown in Fig. 7. The other three processing units of the Packet Processor have essentially the same structure. The processing unit comprises three main components, a processor-1 400, a coprocessor-1 401, and an internal memory unit 402. As said, processor-1 is preferably of a RISC type processor, more preferably of the type known as ARC (by ARC cores, Ltd.). Processor 400 receives data from the Tubular Bus 2430, more particularly from FIFO 270, into its IN FIFO REGISTER 405. The

processor 400 and the coprocessor 401 have each access to the tubular bus. While processor 400 performs mainly operations which are related to the logical analysis including determining the length and type of the packets (management, or data), possible concatenation of packets (in the receiving path) or timing, controlling allocation, and prioritizing the transmission sequences in the transmitting path, and framing or deframing of the data stream, coprocessor 401 do mainly specific tasks, i.e., carrying out the DES and or the CRC. Such tasks are carried out independently by the coprocessor, under the supervision of the processor 400, providing to the coprocessor control signals. When necessary, the processor 400 communicates data that it receives from the Tubular Bus into the coprocessor 401 via the peripheral bus 444 for further processing, or in some cases, when the processor detects that there is no need for any processing in any specific data portion, it allows that data from the Tubular Bus to bypass the processing unit and to be conveyed directly from FIFO-1 270 into FIFO 271. Data that is processed by the processor 400 is conveyed from the Out FIFO Register 406 via the Tubular Bus 2430 into FIFO-2 271. The peripheral bus 444 is a memory mapped local bus through which processor 400 communicates mainly with the coprocessor 401 (primarily for command and control of operation modes), and with the immediate peripherals, like timer 440, the dual port RAM 402 etc. The communication from the processor 400 to the peripheral bus is made through the memory bus 414. The DES Busy and CRC busy status lines 445 and 446 respectively are used

by the coprocessor 401 to indicate to the processor 400 its being in a busy state. The processor further comprises a Status Core register 407, and it also communicates with auxiliary registers 410 via an auxiliary bus 411, which is also an option of the ARC processor. The auxiliary registers, each contain a word which is used for configuration purposes, such as the cache size, the processor 400 ID, etc. The memory of the processing unit comprises three memory sections, an instruction cache memory 415, a dual port SCRAM 402, and an external memory unit. It has been found that with an instruction cache size of 4K bytes, a relatively high hit rate of well above 90% can be achieved by a Packet Processor according to IL 130796, operating as a media access control (MAC) module of a modem. A memory controller 414 controls the communication between the processor 400 and all said sections of the memory. The Instruction Fetch Bus 420 is used by the processor 400 for fetching instructions, and the Load/Store Bus 421 is used for connecting the processor with all peripherals through the Backbone Bus. Specifically, it allows the Load/Store access to the internal SCRAM 402, and the external memory. The communication between the memory controller and the external memory is made through the Sub-Unit Controller 290 over the External Bus 245. The downloading of the internal memory and the cache sections is carried out from a device external to the Packet Processor, via the Backbone Bus 244, the Sub-Unit Controller 290, and the memory controller 414. The Sub-Unit Controller 290 also regulates all kinds of data transfer between the processing unit 251, and the

Backbone Bus 244 or the External Bus 245. The processor 400 of IL 130796 also comprises an internal interrupt interface 417, for handling interruptions. Four hardware interrupt lines are provided from the Interrupt Control Unit 418 to the processor 400. The ARC timer 440 provides a timing clock 441 to the processor 400, and optionally also to other components of the processing unit. More particularly, it serves as a time-out counter, and a real-time clock.

In order to accelerate the data transfer between components inside the Packet Processor or external thereof, the Packet Processor of the IL 130796 is also provided with a DMA (Direct Memory Addressing) controller 448 (Fig. 6) that can enable various of DMA channels. Each channel can transfer a stage of data from one bus or interface to another bus or interface. A DMA channel transfer can be initiated by a processor of a processing unit, or optionally by another controller or interface that has been defined to have that option. Each channel supports a data chaining using a memory located stage of data. The DMA channels are divided into three main data transfer types:

1. A general bidirectional channel for data transfer between:

    (a) The PCI BUS and the Backbone Bus.

    (b) Internal bus and the Ethernet.

    (c) PCI bus and the Ethernet.

2. A unidirectional channel in the received path for data transfer:

(a) From the Tubular Bus to the PCI bus.

(b) From the Tubular Bus to the Ethernet.

(c) From the Tubular Bus to the Backbone Bus.

3. A unidirectional channel in the transmitted path for data transfer:

(a) From the PCI Bus to the Tubular Bus.

(b) From the Ethernet to the Tubular Bus.

(c) From the Backbone Bus to the Tubular Bus.

As said, it is highly advantageous to have at least two processing units in each one of the transmit part and the receive parts of the Packet Processor of the invention of IL 130796. It has been found that the use of a single processing unit in each of said parts, although possible, requires a very powerful processor in the processing unit. The sharing of tasks between two processing units in each part significantly improves the rate and the performance of the Packet Processor. In that case, many tasks can be performed simultaneously by the two processing units in each part. The architecture of this invention is scaleable, allowing the use of more than two processing units in each part may improve even more the performance of the Packet Processor. However, the rate of that improvement may be less than is achieved by the introduction of the second processing unit in each part, due to a possible loading of the internal buses.

In many modem applications the modem may receive data that is actually targeted to another modem. Frequently, such data may be a very large portion of the total data that modems in said applications receive. This is the case, for example, in modems for TV cables. In that case, the modem has to determine whether the data is actually addressed to it or not, or in other words, whether to ignore portions of the data it receives or not. This can be determined only at the packet level, from the section of the packet indicating the destination address. For that purpose, the Packet Processor of the invention preferably further comprises an address filter 272. The address filter according to the invention is located on the Tubular Bus of the receiving part between the first processing unit 251 and FIFO 2 271. It receives packets from Processing Unit-1 251, and it filters out any packet that should be ignored by the Packet Processor. By doing so, it lowers the workload from the entire receive path by dumping any packet received with an address that does not appear in a list of addresses existing in that filter 272. In a particular case when the Packet Processor is used as a MAC module for a modem for TV cables, the list of addresses contains, for example, a set of 16 addresses, each of which is 48 bits wide. All 16 addresses are treated as unicast, i.e. the address filter looks for an exact match between the incoming address and at least one address from the address memory. Each address can be marked as "invalid". The address

filter drops from further processing any packet causing a hit on an "invalid" address.

Of course, the data flowing along the Tubular Bus has a different structure in different sections of the Tubular Bus. According to the embodiment of Fig. 6, the following data structure exists in the indicated sections of the Tubular Bus in Fig. 6:

Section No.	Data Structure
900	Bytes stream with PHY commands
901	MAC frames stream
902	Filtered MAC frames stream
903	Byte stream with FIFO management commands

The structure of the Packet Processor of IL 130796 enables performing the required tasks at a very high rate. The Packet Processor of IL 130796, although suggested herein as an example to be used as a MAC module of a modem, is a general purpose and programmable, as its code can be downloaded from an external source. Therefore, working with the module of IL 130796 is flexible, as it can easily adapt to changes in standards, or data rates. The adaptation of the module to such changes can be simply effected by the modification of the downloaded code.

The Packet Processor also preferably employs the concept of Inband Macro-instructions, where the control of the various stages and co-processors is accomplished by means of Macro-instructions (MIs). The macro-instructions are the in-band commands which are passed between the different Packet Processor components and are embedded in the data stream flowing through the tubular bus. Preferably, the macro-instructions are distinguished from the data by the value of the 9th bit ("1" means Macro-instruction and "0" means data).

The macro-instructions are used to allow control and "message" passing from one unit to another with the ability to keep the close relationship to the data stream. This mechanism allows the Packet Processor to comprise minimum asynchronous signals between the units and to synchronize the data related signals with the data itself, while using FIFOs and buses between the units.

The MIs are generally known to all the components of the Packet Processor. Each component recognizes several MIs and ignores the rest.

According to one embodiment of the invention of IL 130796, each data word in the Packet Processor consists of 9 bits. If the 9th bit is '0' – this word is a data word and is treated as such. If the 9th bit is '1' – this word is a

macro-instruction. The structure of the MI depends on the MI group to which it belongs. Given below is an example of some MIs, their codes, functions and structures. The following Table 1 depicts the global and software groups MIs.

Table 1

8	7 6	5 0
MI Indicator	MI Group	Specific MI code

The following Table 2 defines the MI group codes:

Table 2

Group Name	Code
Software	'00'
General	'10'
FIFO	'01'
Address Filter	'11'

Table 3: MI codes of the general group

MI Name	Code	Description
MI_ABORT	180	Abort operation and restart internal state machine.
MI_PACKET_START	181	Indicating that the next byte is the first byte of a packet.
MI_PACKET_END	182	Indicating that the previous byte was the last byte of a packet.
MI_FRAME_FLUSH	183	Ignore the current frame.

Table 4: MI of the FIFO group

MI Name	Code	Description
MI_FIFO_FLUSH	160	Command to the FIFO to flush itself.

Table 5: MI of the Address Filter

MI Name	Code	Description
MI_ADDR_START	1C1	Instruct the AF (address filter) to start the checking the address from the next address.
MI_ADDR_END	1C0	Notifies the AF that the previous byte was the last address byte.
MI_ADDR_MISS	1C2	A macro instruction issued by the AF notifies the next units on the T-Bus that the address filter did not find an address that matches the last received address. If this occurs the AF stops forwarding data to the T-Bus until the next MI_PACKET_START is received from the T-Bus.
MI_ADDR_HIT	1C4	A macro instruction issued by the AF notifies the next units on the T-Bus that the address filter found an address that matched the last received address. If this occurs the AF continues to forward data to the T-Bus.

Of course, the description above has disclosed only the general structure of the Packet Processor of the invention of IL 130796, which is suitable, for example, to be used in modems for TV cables. The Packet Processor naturally contains other signals and controls that have not been disclosed herein for the sake of brevity. These additional signals are known, and can be easily developed by those skilled in the art. Furthermore, the Packet Processor of IL 130796 can be easily reduced and manufactured in a VLSI form, a preferable form of the Packet Processor of IL 130796.

The performance of the module is best when its tasks are divided between the processing units according to the Venturi Pipeline model.

The Venturi Pipeline model describes a general data communication system, which receives information from a physical layer and transmits information into a physical layer.

The Venturi Pipeline, as is known from classical fluid theory, is a variable diameter pipeline. When fluid is injected into the pipeline, its tangential velocity is higher in the narrow part, and slower in the wider part. The dM/dt, i.e., the mass flow ratio is naturally kept equal along the pipeline, while the same mass of fluid that flows into the pipeline must flow out. The Bernoulli law of fluid flow describes the phenomena wherein the pressure on the pipeline walls decreases, as the velocity of the fluid increases: low

pressure at the narrow part and high pressure at the wide part.

The analogy to a communication system is as follows: The ends of the pipeline describe the input and output streams of the communication system, which are close to the physical layer. The symbol or bit stream at the physical layer level is very high, making the latency cycle required to process the information very short and time-sensitive. As the information is decoded, built into complex streams or packets and deciphered for contents, the size of the information element increases, and at the same time, the information transfer rate is slower. At these stages, the complexity of the processing level increases more and more.

Virtual division of the pipeline into layers, describing the various information elements as they are formed, makes the continuous pipeline a Layered Venturi Pipeline Model for information flow.

Back layers (or group of layers) of information can use a dedicated processor for flow processing. The instructions used by this processor are suitable for the contents and format of the flow at that layer, according to the specific communication protocol and content. At the far sides of the Venturi Pipeline, the processing units can be very small and simple, dedicated processors. The deeper we get into the pipeline (i.e., in our case the Tubular Bus), the more complex the processors must become, in order to support

more computational instructions and to perform a longer series of instructions.

In many cases it is impossible to implement a complete communication device with a single, powerful processor. Although the total processing power is measured in total system MIPS, the real-time requirements of the various levels make it a very challenging burden to manage various tasks from within a single processor, executing a single task at a time. It is sufficient to have two of the layers, both requiring immediate processing within the same short time slot, to make performing the complete task very difficult. When the processing task is divided among several processors as in the MAC module of IL 130796, each running independently, there will be no mutual timing constraints. Furthermore, each change requirement in one of the levels would in most cases not interfere with the other level, making the complete platform very easy to manage and flexible to programming changes.

In the case of the Packet Processor of IL 130796, there is at least one processing unit in each of the receiving and the transmitting parts 23 and 24, respectively. However, in order to comply and exploit the advantages of the Ventury Pipeline Model, the Packet Processor of IL 130796 preferably comprises at least two processing units in each of said two parts. In that case, processing unit-1 51 and processing unit-4 53 perform simple tasks on

small units of data at a very high rate, while processing unit-2 52 and processing unit-3 54 perform more complicated tasks on larger units of data at a slower rate.

In order to optimally utilize the Packet Processor of IL 130796, according to the above described Venturi Pipeline model, there are several options for selecting the processing units:

1) Identical standard processors: this is the simplest option. It is however, the least optimal, as it does not comply with the Venturi model, which, as said, is based on the principle that different processors in the task chain do tasks of varying complexity. Therefore, the processing units should preferably not be the same.

2) Dedicated non-standard processors: using a different type of processor at each stage in the task pipeline is probably the most optimal in terms of silicon size, but the most complex in terms of the development environment: tools (compilers, assemblers, debuggers) are needed to be developed for each dedicated processor.

3) Parameterizable standard processors: This is generally the optimal solution, as it combines the best of all worlds: standard development environment for all processors, and reasonably small silicon area. Such a

processor is the ARC RISC microprocessor that can be parameterized in both the hardware and the Instruction Set (different cache size, different functional stages such as multiply/accumulate, barrel shifters). This selection allows the use of same development tools for all the processors in the task pipeline, while the processors differ in their specific composition according to their location in the task pipeline.

The above description relates to the structure of the Packet Processor according to IL 130796. This description is given herein for the purpose of illustration. The present invention provides a process for processing received and transmitted data streams. This process that can be carried out, by any suitable means, for example, by means of the Packet Processor of IL 130796, or by hardware of different structure meeting the requirements as described hereinafter.

The present invention further relates to a packet processing system. The system comprises a software process for processing received and transmitted data streams, and hardware for carrying out such process.

According to one embodiment of the present invention, the system is used as a cable modem that bidirectionally communicates with the cable system by one port, and with a host by another port. The Host may be a PC or any other computer. The internal cable modem may be located external of the

Host, may be a card that is plugged into the Host, or may be embedded on the Host board. The hardware of the system comprises a modulator-demodulator section and a Packet Processor, preferably having a plurality of processing units. Such Packet Processor may be, for example, the Packet Processor of IL 130796.

For the sake of brevity, hereinafter, unless otherwise specifically stated, it is assumed that the Packet Processor of IL 130796 is used. However, as already explained, the process of the invention can be carried out on any other suitable packet processors as well. According to one embodiment of the invention the hardware of the system comprises:

(a) A Host board;

(b) A modulator-demodulator section;

(c) A Packet Processor;

(d) A Host memory that is located on the Host board;

(e) A Host Bus Interface, such as a PCI interface for communicating with the Host; and

(f) An External memory that is located on the modem's board;

According to one embodiment of the invention the software of the system comprises three main components:

(a) A Microcode that resides in the modem's board and implements the real time tasks of the system;

(b) A Device Driver 866 (Fig. 8) for providing communication between the Host and the Packet Processor; and

(c) A CMMS (Cable Modem Management Service) 867 that implements the high level management, as required by the communication standard used in the cable system.

In addition to the above 3 components, the software subsystem may further comprise a GUI (Graphical User Interface) 868 for providing the user with a mechanism for monitoring and/or for carrying out basic a configuration of the Packet Processor.

The Microcode runs on the modem's board. It mainly carries out the "hard real-time" tasks that are derived from the communication scheme used in the cable system. The main tasks of the Microcode are:

(a) De-framing the incoming data from the modulator-Demodulator and building MAC frames. More particularly, the MAC packets that are carried over the physical layer by a convergence layer (such as MPEG2 in the case of MCNS and DAVIC) are extracted.

(b) Synchronizing the internal timing reference of the modem with the global timing reference as derived from the receiving path.

(c) Testing the transmitting path and the receiving path flowing validity, MAC address, CRC etc.

(d) Analyzing the MAC frame and processing transmission allocation management messages that are generated by the CMTS and are received from the cable network.

(e) Performing the flow encryption in the transmitting part and decryption in the receiving part.

(f) Transferring packets between the Packet Processor and the Host main memory.

(g) Managing and synchronizing the transmission, including transmission requests and re-transmissions (if applicable).

(h) Accumulating statistical information and events as required by the management system.

The Device Driver866 is responsible for keeping the connection between the modem's on-board Microcode, the Operating System 865 of the Host 827, and the CMMS. The main tasks of the Device Driver are:

(a) Managing the initialization process of the modem's board.

(b) Filtering the packets that that should be transmitted and are received from the Host 827 Operating System 865. The filtering is done according to the management system requirements (IP filtering, QoS (Quality of Service) streams, MAC management messages identification etc.).

(c) Forwarding the filtered packets of section (b) to the modem's board.

(d) Filtering the packets that are received from the modem's board according to the management system requirements (IP filtering, QoS streams, MAC management messages identification etc.) before forwarding to the Host 827.

(e) Enabling any general communication between the CMMS 867 and the modem's board.

(f) Accumulating statistical information and events as required by the management system.

(g) Fulfilling the requirements needed for communicating with the Host 827 Operating System 865.

The CMMS (Cable Modem Management Service) 867 is a high-level management service. It implements high level protocols such as SNMP, RSVP, etc., provides configuration to the Device Driver and modem's board according to its internal state machine, and verifies the system integrity. The CMMS 867 main tasks are:

(a) Monitoring system's integrity;

(b) Implementing high-level management protocols (such as SNMP, RSVP, ToD (Time of Day), etc.).

(c) Exporting an updating interface to optional components, such as the GUI 868 (when used).

(d) Managing the registration of high level services, such as QoS and Security. The actual services are implemented at the driver and the Microcode levels).

Fig. 9 illustrates in a block diagram form the basic structure of the process of the invention. The Host processor 816 provides data packets for transmission to the transmitting path 750, and receives processed data packets from the receiving path 740. The Host 827 further communicates with the high layer management of the process. The process of the invention provides to the modulator/demodulator 800 a processed data flow, for example in MCNS format for transmission to the cables network, and receives from the modulator/demodulator 800 a data flow in MCNS format for processing.

The receiving and the transmitting paths 740 and 750 respectively comprise each at least one processing unit, and preferably at least two processing units. Each of these paths further comprises a pipeline, that is generally divided into sections such as 810, 808, and 812, between different components, such as processing units, filter, FIFOs, etc. Over the pipelines data and Macro-instructions progress, while being processed by the processing units. The process of the invention is therefore carried out in combination by means of a real-time software (Microcode), interfacing and

high level software, and hardware (the processing units, FIFOs, pipelines, etc.).

It should be noted herein that some of the components of the system may be carried out alternatively either by software or hardware, as is known to those skilled in the art. For example, the pipeline itself can be a hardware or software pipeline.

The first stage 814 of the transmitting path performs mainly the packets framing, the carrying out of Host 827 interfacing management, and the transmission management. This stage also receives transmission timing information 815 from the receiving path, and conveys transmission timing information 811 to the Timer stage 804, which in turn performs timing of all the transmission process, and timing analysis. Stage 803 of the transmitting path performs mainly encryption of the data stream, before conveying the processed data via section 801 of the pipeline to the modulator/demodulator section 800 for transmission.

Stage 807 receives a data stream from the modulator/demodulator section 800, deframes the packets, and checks the CRC validation. It further provides timing information 806 to the timer stage 804.

Filter 809 performs packet filtering. More particularly, it compares the address that is included in a header of a processed packet with a list of addresses that are defined as relevant to the Host 827. If the address in the header is found to be irrelevant, the processing of this specific transfer lapses, and the irrelevant received packets are erased from the process. Packets with a relevant destination are allowed to continue to the Decryption stage 813. Stage 813 performs management analysis, interfacing with the Host 827, and decryption of the data. It also conveys transmission timing information 815 to stage 814, for allowing this stage to manage the transmission slot timing allocation.

The Timer stage 804 performs timing analysis and track keeping (i.e., counting the elapsed time). It also provides start/stop transmission control 817 to stage 803.

It should be noted that the invention is primarily based on the use of two separate pipelines, one in the in transmitting path and a second in the receiving path, each pipeline is divided into segments (for example segments 808 and 812 in the receiving path, and segment 810 in the transmitting path). The use of pipelines, in association with plurality of processing units and the process of the invention as described hereinafter, enables a simultaneous performance of many tasks. It should be noted that in Fig. 8,

the block diagram does not intend to indicate a serial performance, as many of the tasks are performed simultaneously.

Fig. 8 illustrates the structure of the process of the invention, particularly the low-level process that is carried by the microcode. As said, the microcode operates on the modem's board, and particularly on the Packet Processor. It interfaces with the Device Driver on one hand and the physical layer of the Packet Processor on the other hand. The microcode is divided into small functional modules, each module performing a distinctive task. The microcode communicates and synchronizes by means of the receiving and transmitting Pipelines, and Macro-instructions.

As mentioned, the Packet Processor of IL 130796 can contain different number of processing units. The process of the invention is designed such that the tasks can be distributed to any number of processing units. The tubular buses and Macro Instructions are integral parts of the process design. In the structure of Figs. 8 and 9, if more than one stage is implemented on a single processor, the stages communicate via a software pipeline (i.e., a simple queue mechanism that is implemented by software). Whenever processing of two adjacent stages in Figs. 8 and 9 is carried out by two different processors, the processors communicate via hardware pipelines, in this case via a section of the tubular bus, and FIFOs. Therefore, it should be noted that the pipelines may be, in some cases,

entirely hardware pipelines, in other cases entirely software pipelines, and in most typical cases the pipelines comprise segments of software pipelines and segments of hardware pipelines.

The receiving path 740 receives a flow of byte stream from the Demodulator of the modulator/demodulator section 800, de-frames it, and forwards the relevant packets to the Host 827. In addition to this main stream processing function, the receiving path keeps track with the network's Global Timing, and forwards received transmission information to the transmitting path 750, particularly to its transmission management stage 714.

The byte stream reception stage 702 is the first software stage that manages the data received from the physical layer, i.e., the modulator of the modulator/demodulator section 800. The byte stream reception stage reads the data received from the physical layer interface (in the case of the Packet Processor of IL 130796 the PHY interface), and performs a PMD (Physical Media Dependent) sub-layer synchronization. For example, in the case of MCNS and DAVIC, stage 702 finds the MPEG2 packet header. Stage 702 reads the received data at a relatively constant rate. The average reading rate is consistent with the network's transmission rate.

In a typical case, the PMD sub-layer comprises a header and payload. The time required for analyzing the header may be longer than the time

required to forward the payload, once a decision regarding the payload is made. The stage complies with the following constraints:

1. The maximum time required for analyzing the header and payload is less than or equal to the time required for the PHY Interface 764 to receive the entire PMD sub-layer packet (header and payload).
2. The analysis of the header by this stage must not cause the pipeline 701 between the physical layer interface (not shown) and stage 702 to fill up.
3. The forwarding of the payload has to be faster then the input byte rate to stage 702. This assures that even if the header analysis causes the pipeline 701 to accumulate bytes, the forwarding of the payload empties it.
4. When forwarding an average payload, stage 702 in most cases empties the pipeline 701 entirely, before the next header is being received. This assures that the next header enters into an essentially empty pipeline 701. Not emptying the pipeline 701 might cause the pipeline 701 to slowly fill up, causing an overflow.

The task of stage 702 may be synchronized with the PMD's received packets by waiting for a specific Macro Instruction from the PHY Interface 764. The Macro Instruction is inserted by the PHY Interface 764 into the pipeline upon detection of a special "frame sync" signal from the modulator/demodulator 800. More particularly, the "frame sync" is a signal that is generated by the PHY interface 764 in order to synchronize the

Packet Processor with the PMD packets stream. The "frame sync" signal is received only when full packets are forwarded to the pipeline 701.

The packets are forwarded from stage 702 into the pipeline 730. The deframing stage 703 reads PMD sub-layer packets from pipeline 730 and extracts from it MAC frames. The de-framing stage 703 rebuilds the MAC frames from the PMD sub-layer frames and dumps the PMD specific information (header and packet validity) that becomes unnecessary. Once the MAC layer frame is identified, stage 703 forwards the frames to its output pipeline 760. Before stage 703 outputs the first octet of the frame to pipeline 760, it places a Macro Instruction indicating the event to the rest of the tasks down the pipeline.

The Initial Packet Analysis stage 704 receives the stream from pipeline 760. This stage analyzes the MAC header and performs the MAC payload validation (CRC checking). More particularly, this stage performs the following tasks:

(a) Verification of the MAC header's validity: In most advanced communication schemes the MAC header carries its own validation information (CRC or checksum). If the test fails, the stage dumps the frame, i.e., does not forward it to the next stage, and injects a special Macro-Instruction indicating the event.

(b) Verification of the MAC frame's validity: Stage 704 further activates a validation mechanism over the MAC frame's payload. If the validation fails, the stage initiates a special Macro Instruction indicating the event to the rest of the stages down the pipeline.

(c) Identifying special timing synchronization messages: In a TDMA (Time Division Multiple Access) scheme, all the nodes of the network are required to be synchronized by a single timing reference. A MAC management message is therefore identified by this stage, and the timing information that is acquired from this message synchronizes a transmission synchronization hardware component, the Timer stage 712 (stage 277 of IL 130796, Fig. 6 of the present application). More particularly, when stage 704 identifies a timing synchronization message, it forwards it to the Timing Analysis stage 713, and not to the Packet Filtering Stage 705 down the pipeline.

The Timing Analysis stage 713 analyzes the timing synchronization MAC management messages that are received from the Initial Packet Analysis stage 704, and programs the Timer stage 712. If the timing synchronization information is carried in a MAC frame's payload, stage 713 also verifies the payload's validity (CRC or checksum). After the validation, stage 713 analyses the received information, and programs the Timer 712 accordingly. Another task of stage 713 is to keep track of the Timer 712 updating-rate. If the rate does not comply with the protocol specification, it generates an

event signal notifying the Host 827 that the system is out of synchronization.

It should be noted that the reasons for carrying out the timing analysis in two separate stages, stage 704 and stage 713 are:

(a) This structure minimizes the latency from the time of receiving timing information, to the time until the Timer 712, generally a hardware component, is updated.

(b) This structure enables the implementing of these functions by two separate hardware components, such as by two separate processors.

(c) This structure enables the carrying out of the more complex analysis beyond the address filtering stage 705 down the pipeline. More particularly, this stage analyzes only what must be analyzed at this stage, and postpones what can be analyzed later.

The Packet Filtering Stage 705 tests the received packet's MAC address. If the test is successful, the packet is forwarded to the next stages down the pipeline. Otherwise, stage 705 dumps the packet. More particularly, in the case of failure, stage 705 does not forward the packet to pipeline 780. The Packet Filtering stage 705 is a slave of the Initial Packet Analysis stage 704. When it receives a Macro-Instruction indicating the beginning of a packet, it starts accumulating the data read from the pipeline 770. It then waits for the Macro-Instruction indicating the beginning of the MAC

address. Then, the stage 705 compares the address with a list of addresses stored in it (or optionally elsewhere), and makes a decision. If the decision is to forward the packet, it first forwards the accumulated data received before the MAC address, and then the rest of the packet. If however, the decision is to dump the packet, the stage dumps both the accumulated data and the rest of the packet that is read from the pipeline 770. The Packet Filtering stage 705 utilizes a set of explicit MAC addresses that are defined as valid, and may optionally also utilize a set of rules. In that case, the Packet Filtering stage 705 applies rules as inputs to a search algorithm, which in turn uses a MAC address as found for carrying out verification and for making a decision accordingly. The Packet Filtering stage 705 is configured by the Host 827, which provides to it a list of valid addresses, and therefore determines MAC addresses that should be accepted and MAC addresses that should be rejected.

The Advanced Packet Analysis stage 706 is a decision point. This stage analyses the MAC header and the payload (if necessary) and forwards each packet to either the Security stage 707, the Flow Transmission to Host stage 708, or to the TX Information Filter stage 710. Once the decision is made, the packet is forwarded to the appropriate pipeline 790, 801, or 810. If the packet is forwarded to the Flow Transmission to Host stage 708, it will (eventually) be forwarded to the Host 827. In this case, stage 706 provides the Host Interface Management stage 711 the new packet's attributes,

particularly type and size. This enables the Host Interface Management stage 711 to manage the DMA (Direct Memory Access) that is performed by the DMA stage 709, and to forward the packet to the appropriate memory buffer in the Host 827. The Advanced Packet Analysis stage 706 further verifies the MAC frame validity (CRC or checksum). While the data is flowing through this stage it calculates the validation value. At the end of the frame, it verifies the received verification value with the calculated one. If the verification test fails, it signals the failure to the following stages:

(a) The TX Information Filter stage 710, if the decision is to forward the frame to this stage. In that case, the Advanced Packet Analysis stage 706 injects a Macro-Instruction to the pipeline segment 790 indicating that the packet validation has failed.

(b) The Host Interface Management stage 711, if the decision is to forward the frame to the Security stage 707, or to the Flow Transmission to Host stage 708. The signaling to one of said stages is preferably carried out by means of a special hardware (interrupt) or software (function call or shared flag), rather than by using the pipeline architecture (i.e., a Macro-Instruction).

Stages 710 and 711 are also responsible for performing the necessary steps in order to discard the frame in the case of invalidity.

The packet validation (CRC calculation) that is performed by this stage can be carried out either by hardware or by software. In the case that the Packet Processor comprises one or more hardware components for implementing the validation process, the software process of the invention activates the said hardware components. In the case of using the Packet Processor of IL 130796, each of said hardware component/s is a co-processor that is contained within a processing unit. In the case that such a hardware component does not exist, the actual CRC calculation is alternatively carried out by means of software.

The Security stage 707 performs a reception security process, generally decryption, of secured packets. The stage receives the necessary security information (such as decryption key) from the Host 827, that implements a high level security protocol. The actual security process may be handled by a hardware component, or by means of software, depending on the structure of the Packet Processor used. The security module 707 receives commands from the Advanced Packet Analysis stage 706, informing it of the type of security and parameters used.

The Security stage 707 is optional, since not all protocols or network schemes implement security features such as decryption.

The Flow Transmission To Host stage 708 receives data from either the Advanced Packet Analysis stage 706 or the Security stage 707. The stage removes Macro-Instructions that are embedded in the data flow from pipeline 810, and forwards the rest to the DMA (Direct Memory Access) stage 709. The DMA stage 709 in turn forwards the data to the Host 827 memory, as instructed by the Host Interface Management stage 711. It should be noted that the use of the DMA stage 709 is preferable in order to accelerate the transfer rate of the processed packets to the Host 827. However, said stage is optional as the data may be transferred in a conventional manner with no use of DMA.

In a TDMA broadcasting network it is typical to use a MAC management message that contains transmission allocation information for more than one node. Such use saves both complexity and overhead, and hence improves the network overall performance. The TX Information Filter stage 710 filters the management messages and determines from the allocation information whether to accept or reject the message. Stage 710 is however optional, since some network schemes do not follow this procedure.

A single node has no use for information concerning other node's transmission allocation. In a single node application, stage 710 extracts therefore from the transmission allocation management message only the transmission information concerning this node/modem. In order to perform

its task stage 710 needs to receive information from the driver/CMMS, indicating which transmission allocations to grant and which to reject.

When a transmission allocation management message is received, as is commonly used in MCNS/DAVIC, it is forwarded to the TX Information Filter stage 710. This stage filters the desired transmission allocation information and builds a data structure that is required by the Transmission Management stage 714 of the transmitting path 750. When the data structure is ready and stage 710 receives an O.K. signal from the Advanced Packet Analysis stage 706, by means of a Macro-Instruction indicating that the packet was properly received, it signals the Transmission Management stage 714 that it can use the information. The signaling is carried out by hardware means, such as a hardware interrupt when the said two stages 706 and 714 are implemented by two different processing units. Alternatively, the signaling can be carried out by means of software, such as by a flag or a function call.

It should be noted that the two tasks dealing with the transmission management, i.e., the task of the receiving path and the task of the transmitting path do not communicate using the "normal" pipeline architecture. They rather communicate via a shared memory. This technique allows the task originated in the receiving path to build a complex

data structure that eases the task of the Transmission Management stage 714 of the transmitting path.

In order to shorten the response time from the time of receiving the transmission allocation MAC management message, to the time of finishing the message analysis by the Transmission Management stage 714, the Transmission Management stage 714 can start analyzing the received information before it receives an O.K signal notifying that the TX Information Filter has finished updating the memory, but it uses the data from said memory only after an OK signal is received.

The Host Interface Management stage of the receiving path 711 is responsible for the memory management between the Host 827 and the Packet Processor. The Host 827 allocates buffers in its own memory for the received packets, and notifies the Host Interface Management stage 711 about each buffer's status (location, size, etc.). This stage manages a DMA channel that has been previously programmed to forward data from the Packet Processor to the Host's memory.

The Host Interface Management stage 711 of the receiving path has three sources of input:

1. The Advanced Packet Analysis stage 706 notifies stage 711 when a new packet is ready to be forwarded to the Host 827. The Advanced Packet

Analysis stage 706 further informs the Host Interface Management stage 711 about the new packet's type and length. In addition, when the Advanced Packet Analysis stage 706 identifies an error, it is signaled to the Host Interface Management stage 711.

2. The DMA 709 notifies the Host Interface Management stage 711 when it finishes transferring a stage of data to the Host's memory. The DMA 709 also notifies the Host Interface Management stage 711 when it detects errors. When DMA is not used, said operations are carried out by the Flow Transmission to Host Stage 708.

3. The Host 827 notifies the Host Interface Management stage 711 about memory buffers (in its memory) that are ready to receive data.

The Host Interface Management stage 711 keeps track of the available buffers received from the Host 827 and programs the DMA 709 to transfer a packet (or a fragment of a packet) to the Host's memory. When the DMA 709 notifies stage 711 that the transfer has ended (the buffer is full, or contains a full packet), stage 711 notifies the Host 827 that it can use the buffer. If an error is detected in the packet reception, stage 711 notifies the Host 827 that a packet has been improperly received. It then may reuse the memory buffer.

The transmitting path 750 is responsible for the reception of a byte stream from the Host 827, for the transmission of data to the network, and for the framing of the packets in a MAC frame format.

The Transmission Management stage 714 is the heart of the transmitting path. The main tasks of stage 714 are the defining of the transmission order and performing timing control.

The Transmission Management stage 714 defines the order and timing of the transmitted packets, and provides this information to both the Host Interface Management stage 715, and to the Framing stage 721. The Transmission Management stage 714 also receives transmission information from the TX Information Filter 710. This information defines the type of packets that can be transmitted in any given time-slot, and the time-slot timing. It also receives from the Host Interface Management stage 715 a list of available/ready packets for transmission. The stage's task is to match each packet with an appropriate time-slot for transmission, to generate a commands queue to the Host Interface Management stage 715, and to the Framing stage 721, and to provide transmission timings information to the Timer stage 712.

If a certain packet can not be transmitted, for example because it does not have an appropriate time-slot, the Transmission Management stage 714

submits a transmission request (if the protocol used allows it) to the upstream transmission allocation authority (e.g., the CMTS).

When the Transmission Management stage 714 has decided about the packet order, it generates instructions to the pipeline that are read by the Framing stage 721. The pipeline therefore contains a set of commands regarding as to how the packet should be framed. The commands are processed by the Framing stage 721. Such commands include information regarding the type of the MAC layer header that has to be inserted, the security properties, etc. The Framing stage 721 acts upon instructions that are read from a pipeline. Said instructions are inserted to the pipeline by stage 714. Stage 721, forwards the data to the Security stage 718 or to the Byte Stream Transmission stage 719, and inserts a Macro-Instruction instructing the said stages as to how to perform their tasks.

The Transmission Management stage 714 keeps track of its posted transmission requests. If a certain request has not been fulfilled, the stage has to retransmit the request. If the request can still not be fulfilled (this can be detected by either an explicit negative acknowledgment or by not responding to the request), this stage notifies the Host 827 about the error.

Since the CATV network is not a reliable media, most of the typical protocol schemes present an acknowledge/retransmit mechanism. The Transmission

Management stage 714 receives with the transmission information from the TX Information Filter Stage 710 a positive acknowledgment or a negative acknowledgment for transmitted packets. Then, the Transmission Management stage 714 decides, based on a set of rules predefined by the Host 827 or the protocol, whether a certain packet has to be retransmitted or to be returned to the Host 827 with an error indication. The decision is passed to the Host Interface Management stage 715 as a release/fetch Macro Instruction.

A part of the decision process of the Transmission Management stage 714 is determining which packet should be transmitted and in which time-slot. The result of the decision is a list of transmission timings. The Transmission Management stage 714 programs the Timer 712 for the next transmission time, when necessary. When a transmission time has reached, the Timer 712 signals the PHY interface 764 and the Transmission Management stage 714. The PHY Interface 764 starts to transmit the data while the Transmission Management stage 714 makes sure to continue programming the Timer 712 with the next transmission timing.

The Host Interface Management stage 715 of the transmission path 750 gathers information regarding packets that are ready for transfer from the Host 827, and notifies the Transmission Management stage 714 of the current status. The Host 827 and the Packet Processor share a section of

memory in which the driver builds a descriptive data structure for each packet it wishes to send. The data structure contains information regarding the actual memory buffer containing the packet (such as location and size), and information describing the packet in terms of the current protocol scheme used (such as DAVIC or MCNS). The Host 827 can optionally use multiple queues for temporarily storing the transmitted packet, in order to provide QoS (Quality of service) flows for different data flows, such as, Constant Bit Rate – CBR – for voice service, and Best effort – BE – for HTTP services.

The Host Interface Management stage 715 can poll the shared memory and identify ready packets. It then signals the Transmission Management Stage 714 which packet is ready for transmission. Stage 715 might signal that several packets are ready, as it is the Transmission Management stage 714 responsibility to serialize the packet stream through the modem.

The Host Interface Management stage 715 further receives instructions from the Transmission Management stage 714 regarding the order in which the packet should be read from the Host 827 into the Packet Processor. Once the Transmission Management stage 714 has defined the reading order of the packets, it signals this information to the Host Interface Management stage 715. Stage 715 then programs the DMA component 722 and monitors the Flow Reception From Host stage 717 for insuring their proper operation. The

DMA stage 722 is optional. If it does not exist, Flow Reception From Host stage 717 reads directly from the Host 827 memory.

Assuming that the DMA stage exists, the Flow Reception From Host stage 717 reads data from the DMA component 722 and forwards it to the Framing stage 721. When stage 717 detects that a packet has been fully read, it notifies the Host Interface Management stage 715. If the Host Interface Management stage 715 detects a condition of data transfer error, it notifies the Flow Reception From Host stage 717. Upon such a signal, the Flow Reception From Host stage 717 reads all the data from the DMA 722 and dumps it. More particularly, in the case of error, the data is not forwarded to the Framing stage 721. The Flow Reception From Host stage 717 also inject to the pipeline a Macro-Instruction notifying the next stages that the packet is not complete and that they should also dump all remaining parts of it. Once stage 717 identifies the beginning of a new packet coming from the DMA 722, it continues to forward data to the Framing stage 721.

The Framing stage receives an instruction queue (a pipeline with Macro Instructions) from the Transmission Management stage 714 which contains a set of commands for each packet. The Framing stage 721 is responsible for the forwarding of the complete data flow to the Security stage 718 or the Byte Stream Transmission stage 719. The complete data flow includes the

MAC layer header, the verification information (i.e. CRC), and of course the payload.

In addition to the transmitted information, header, payload and verification, the Framing stage 721 generates Macro Instructions to the Security stage 718, or the Byte Stream Transmission Stage 719 regarding the security parameters (if applicable) and the transmission parameters. The transmission parameters can include "high" level information such as the packet length, or "low" level information such as the modulation type, transmission scrambler type, etc. The Macro-Instructions are handled by the Byte Stream Transmission Stage 719, by means of hardware components.

The Security stage 718 performs a transmission security process, particularly encryption of the packets. Stage 718 receives the necessary security information, such as encryption key, from the Host 827 that implements the high level security protocols. The actual security process is preferably handled by a hardware component. Alternatively, the security process can be carried out by software. In the case of using a Packet Processor of IL 130796, the security is handed by a co-processor that is responsible for the encryption, and is controlled by a processor, both being located within a processing unit. The security stage 718 receives commands from the Framing stage 721 instructing it what type/parameters of security to use. In any case, the security stage 718 is an optional software or

hardware component, since not all protocol/network schemes implement security features.

This following table maps the process design as defined above to the requirements derived from the MCNS standard specification. The MCNS standard is known to those skilled in the art, and therefore is not explained in detail herein, for the sake of brevity.

STAGE	MCNS requirements
702	MCNS PMD is an MPEG-2 packet with a special MCNS PID (Program ID). This task analyzes the MPEG-2 header and filters in those MPEG-2 packets that belong to the DOCSIS system. If synchronization with the PMD layer has been lost, for example, as the MPEG-2 header has not been recognized on time, the stage injects a Macro-Instruction to the pipeline to notify the rest of the stages.
703	MAC frames extraction from the MPEG-2 packet. This stage reads the MPEG-2 packet header and looks for the PUSI (Payload Unit Start Indicator) field in the header. The stage does not forward the MPEG-2 header. As long as the stage is not locked on the MAC frames stream, it does not forward any data to stage 704. Once a PUSI was recognized, the stage can lock on a MAC frame stream. The stage now starts forwarding the MAC frame to stage 704. Before each MAC frame, the stage injects a "frame start" Macro-Instruction as an indication to the rest of the stages.

STAGE	MCNS requirements
704	This stage carries the 1st level analysis. It analyzes the MAC header in order to recognize: 1. The MAC timing header: In the downstream flow, the timing header indicates a MAC Management Message (hereinafter: "MMM") that needs to be forwarded to stage 713. Otherwise data is forwarded to stage 705. 2. The MAC header length in order to correctly operate and verify the HCS (Header Checksum) and CRC validity. If one of the tests (HCS or CRC) fails, a Macro-Instruction is injected in order to notify the rest of the stages down the pipeline. When the end of the header is reached, the stage injects an "address start" Macro-Instruction to indicate stage 705 that the next byte is the first byte of the Ethernet address. Six bytes later it injects an "address end" Macro-Instruction.
713	This stage verifies that the received message from stage 704 is a SYNC MMM by testing the LLC fields (LLC type 1 – SYNC and version 1 (MCNS specific)). Then it extracts the "CMTS (Cable Modem Termination System) time-stamp" field from the message and loads it to the Timer hardware stage 712. This stage is also responsible for the SYNC update time-out. Each time a "CMTS time-stamp" is loaded to the Timer 712, a time-out counter is being initiated. If a SYNC message is not received within the required time-out – it notifies the Host 827 about the event.
705	The stage filters in only those packets carrying an allowed DA (Destination Address). This includes the Ethernet and MCNS broadcast, CPE (Client Premises Equipment), the CM's (Cable Modem) unicast, and any multicast data obtained during the Cable Modem operation. The stage forwards all the data from the "frame start" Macro-Instruction until a decision is made. If the packet is rejected, the "address end" Macro-Instruction is replaced by an "address missed" Macro-Instruction, which indicates that the packet has failed the address filter check.
706	This stage performs the second level analysis of the MAC frame. This stage is a decision point. It forwards data to: Stage 707 – if a MAC frame contains a BPI (Baseline Privacy Interface) header with the BPI_ENABLE = 1; or Stage 710 – if the frame is a MAP MMM (A MAP is a type of a message in MCNS); or Stage 708 – otherwise.
707	This stage performs the operation of the DES decryption. It also analyses the BPI_DOWN (DOCSIS Specific) extended header. It also carries the program of the DES decryption HW component (when exists) with the key and the initial vector associated with the received SID (Service ID). The SID itself is embedded in the BPI extended header.

STAGE	MCNS requirements
708	This stage performs tasks that are not related to the MCNS. It removes unnecessary Macro-Instructions that are embedded in the data stream and forwards the rest to the DMA stage 709 (or dors it itself, when a DMA is not in use).
710	The MAP MMM is forwarded to this stage. The stage filters each IE (Information Ethernet) in the MAP (common knowledge to those familiar with the DOCSIS/MCNS specifications) and decides whether it should be inserted to the data structure 731 that is shared with stage 714. The data structure contains a MAP header summary and the compiled IEs. When the stage detects that the packet is received properly (the CRC test passed), it signals stage 714 that it can use the information stored in the shared data structure.
711	This stage performs tasks that are not related to the MCNS. Its sole purpose is to manage the transfer of a packet from the board to the Host's memory.
714	This stage is the heart of the transmitting path 750. It is responsible for the following issues that are defined in the MCNS specifications: ➢ Submitting the transmission opportunities requests for the packets waiting to be transmitted; ➢ Fragmenting and concatenating packets when applicable; ➢ Defining the packets' transmission order according to the MAP MMM; ➢ Re-transmitting packets and transmission requests (when applicable), and managing the contention resolution process (MCNS specific); and ➢ Programming the Timer hardware component 712 for transmission accuracy.
715	This stage performs tasks that are not related to the MCNS. Its sole purpose is to manage the transfer of packets from the Host's memory to the Packet Processor.
717	This stage performs tasks that are not related to the MCNS. It reads data from the DMA stage 7, signals stage 715 when a full packet has been read, and injects Macro-Instructions when necessary.
721	This stage builds an MCNS MAC frame according to instructions from stage 714. It adds the MAC header (including any necessary extended headers) to the stream, and injects Macro-Instructions for instructing stages 718 and 719 how to operate.
718	This stage carries the operation of the DES encryption. It also programs the DES decryption hardware component (when exists) with the key and with the initial vector associated with the main SID.

STAGE	MCNS requirements
719	This stage controls the programming of the interface to the modem, such as the PHY interface, according to each IUC's (Interval Usage Code, MCNS specific) burst profile.

As said, the process of the invention can be carried out on different hardware platforms, which may have different structures, as described above. For example, the process of the present invention can be carried out on the platform of the Packet Processor of IL 130796. In that case, the received path 740 is implemented by Processing Unit-1 251 (Fig. 6) and by Processing Unit-2 252. The transmitting path 750 is implemented by Processing Unit-3 253 and by Processing Unit-4 254. Processing Unit-1 251 and Processing Unit-4 254 carry out the low level, "hard" real-time tasks that are related to the transmission/reception to/from the modulator/demodulator 800. Processing Unit-2 252 and Processing Unit-3 253 are the back processing units, which implement the more complex, less real-time tasks. The following table maps each of the stages defined above, and with reference to Figs. 8 and 9 to an actual hardware processing unit.

Processing Unit	Implements stages...
1	702, 703, 704 and 713 (stage 705 is implemented by a hardware component, the address filter 272 – Fig. 6).
2	706, 707, 708, 711 and 710
3	714, 715, 717 and 721
4	718 and 719

The Packet Processing platform that is required for carrying out the process of the present invention can include any number of processors. Each task may run on a different processor. Several tasks simultaneously operate on different sections of the data flow. Each task analyzes and manipulates a specific section, and it forwards the manipulated data to the next task in line.

For example, let assume that task A implements a very simple and deterministic state machine. Task A reads in a constant rate data from the data flow, and forwards it to the next task, task B. Task B has to analyze the header of a packet, a job that might take longer then the input rate from task A. This forces task A to synchronize with task B and to forward a byte to it only when task B is ready. The use of a FIFO (pipeline) between the tasks eliminates the need for synchronizing the two tasks. Task A can forward a byte to the FIFO whenever it has a byte ready, and task B can read a byte whenever it is ready to receive such byte.

This mechanism disjoints the tasks to a point where one task can be changed with no need for changing any of its adjacent tasks.

While using a pipelining architecture, two adjacent tasks do not handle the same, or generally even not a consequent byte. This requires precise synchronization between any two tasks. For example, if the first task tries

to lock on the beginning of a MAC frame, it must notify the second task where the frame starts.

This solution is provided by widening the data pipeline to hold both data and meta-data (Micro Instructions) for synchronization and control purposes. A ninth bit is presented in each octet. The ninth bit indicates whether the octet is a part of the data stream, flowing between tasks, or a Macro-Instruction that is injected to the stream for management and synchronization purposes.

As shown, the invention provides a flexible process and system for packet processing. The process and system can be adapted to work with different communication protocols, and can be adapted relatively easy to changes and updates in communication protocols, as these adaptations, according to the invention generally involve software changes only. This is particularly important, as the method and system of the invention are designed to operate in fast-developing environments, such as the environment of communication over TV cables. Moreover, this advantage is provided with no reduction in the processing speed, in comparison to the prior art.

While some embodiments of the invention have been described by way of illustration, it will be apparent that the invention can be carried into practice with many modifications, variations and adaptations, and with the

use of numerous equivalents or alternative solutions that are within the scope of persons skilled in the art, without departing from the spirit of the invention or exceeding the scope of the claims.

CLAIMS

1. A process for carrying out the processing of received and transmitted packet streams comprising:

a. Providing two sections, a transmitting section and a receiving section, each section comprises of at least one processing unit;

b. In the receiving section:

b.1. Receiving a data stream of packets from a demodulator section, passing the stream through a receiving pipeline, while processing the data by said at least one processing unit of the receiving section, processing tasks are performed simultaneously when possible;

b.2. When all processing tasks are completed, conveying the processed packets to a Host;

c. In the transmitting section:

c.1. Receiving a packets stream from a Host section, passing the stream through a transmitting pipeline, while processing the packet stream by said at least one processing unit of the transmitting section, processing tasks over the data stream are performed simultaneously when possible;

c.2. When all processing tasks are completed, conveying the processed packets into a modulator for transmission;

2. A process according to claim 1 for carrying out the processing of received and transmitted data streams made of packets in a Packet Processor, the process comprising:

A. a receiving process comprising:

a. Providing a receiving pipeline, and at least one processing unit;

b. Receiving into said pipeline a stream of packets from a demodulator section, each packet mainly comprises a header and a payload section;

c. Identifying the header of the received packet;

d. Analyzing the header of the packet for checking the header validity and for extracting from it timing information (when exists), and addressing information, while forwarding the packet downstream the pipeline.

e. Simultaneously with the analysis of the header, conducting a validity check of the payload; if the validity check of the payload is found valid, forwarding the payload downstream the pipeline;

f. Optionally, if the payload is encrypted, carrying out decryption of the payload; when the decryption is completed, forwarding the payload towards the Host;

g. Simultaneously with one of steps d-f, comparing the addressing information as extracted in step d with a list of valid addresses;

h. Optionally, if any time during the performance of steps d-f, at least one of the following failures occurs:

- the validity check of the header (step d) fails; or
- the validity check if the payload fails (step e); or
- the addressing information in the header is found to be different from any address in said list of valid addresses (step g); or
- a packet could not be completely received;

Terminating the processing of the packet, dumping the information relating to the said packet, and initiating an error message that corresponds to the type of failure.

B. A transmitting process comprising:

a. Providing a transmitting pipeline, and at least one processing unit;

b. Receiving by the pipeline a payload from a Host, and at least a destination address; forwarding the same in the pipeline;

c. Framing the payload, including at least creating an associated header, and simultaneously with said header creating, adding validity information to said header; associating the header with the payload and forwarding in the pipeline;

d. Optionally, encrypting the payload;

e. Simultaneously with steps b-d, preparing a proper transmission timing slot for the transmission of the packet;

f. Any time during the performance of steps b-e, if an error occurs, terminating the processing of said packet for transmission, dumping

the information associated with said packet, and initiating an error message notifying the type of the error;

g. If no error is detected in step f, forwarding the resulting packet to a modulator for transmission.

3 A method according to claim 2, wherein the Packet Processor comprises:

A. A receiving section comprising:

a. A receiving tubular bus;

b. At least one processing unit connecting between segments of the tubular bus;

c. One FIFO before and one FIFO after any processing unit on the tubular bus;

d. Optionally at least one address filter for comparing an address extracted from a header of a received packet with a list of valid addresses;

B. A transmitting section comprising:

a. A transmitting tubular bus;

b. At least one processing unit connected between segments of the tubular bus;

c. One FIFO before and one FIFO after any processing unit on the transmitting tubular bus;

C. A PHY interface for receiving data from a modulator/demodulator unit and conveying it to the receiving tubular bus, and for receiving

data from the transmitting bus and conveying it to the said modulator/demodulator unit for transmission;

D. A Host interface for receiving data from a Host and conveying it to the transmitting tubular bus, and for receiving data from the receiving tubular bus and for conveying it to the Host;

E. At least one bus for conveying management information between different components of the Packet Processor; and

F. Timing and control means for administering the operation of the Packet Processor, and particularly for providing the allocation of transmission slots for the transmitting of packets.

4. A system for carrying out the processing of received and transmitted packet streams comprising:

a. a modulator/demodulator section for transmitting and receiving packet streams respectively;

b. A Packet Processor comprising a transmitting section for preparing packets for transmission and a receiving section for extracting the data from a received packet stream, each of said transmitting and receiving sections comprising at least one processing unit, a tubular bus connecting the processing units, and a FIFO before and a FIFO after each processing unit on the tubular bus;

c. A Host for receiving processed data from the receiving section of the Packet Processor, and for conveying data for transmission into the transmitting section of the Packet Processor; and

d. A process according to claim 1 for processing received and transmitted packet streams.

5. A system according to claim 4 wherein the process is a microcode carrying out real-time processing in the Packet Processor.

6. A system according to claim 5, wherein the microcode part of the process comprises:

A. A receiving section comprising:

a. A one way pipeline made of segments, said pipeline receives data streams from the modulator/demodulator section;

b. MacroInstructions embedded with the data flowing over said pipeline, for conveying commands between different stages that are connected to the pipeline;

c. A Byte Stream Reception stage for receiving framed packets from the pipeline, and extracting PMD frames from the data stream and forwarding the resulting frames into a De-Framing Stage via the pipeline;

d. A De-Framing stage for receiving data from the Byte Stream Reception Stage via the pipeline, extracting a MAC Frame, and

dumping PMD header and validity data; forwarding the resulting stream into an Initial Packet Analysis Stage;

e. An Initial Packet Analysis stage for receiving data from the De-Framing stage via the pipeline, analyzing the said MAC header, forwarding timing information into a Timing Analysis stage, carrying out a validity check on the MAC header, and forwarding the resulting packet stream into the pipeline;

f. A Packet Filtering stage for comparing addressing information that is extracted from the MAC header with a list of valid addresses; if the result of the comparison is negative, dumping the current packet; if the result of the comparison is positive, forwarding the stream into an Advanced Packet Analysis stage;

g. An Advanced Packet Analysis stage for carrying out validity check on the payload, forwarding transmission timing information into a TX Information Filter, and forwarding the resulting payload into a Flow Transmission to Host stage, or optionally into a Security stage; and

h. A Flow Transmission to Host stage for receiving a payload, and conveying same into a Host;

i. An optional Security stage for decrypting the payload, if the payload is encrypted.

B. A transmitting section comprising:

a. A one way transmitting pipeline made of segments, said pipeline receives unframed payloads packet information from the Host;

b. A Flow Reception From the Host stage for receiving unframed payload packets information from Host via the transmitting pipeline, and forwarding the same into a Framing stage via the transmitting pipeline;

c. A Framing stage for constructing a MAC header, combining validity information with the header and payload, and forwarding the resulting framed data packets into a Byte Stream Transmission stage or optionally a Security stage via the transmitting pipeline;

d. A Security stage for optionally receiving framed data packets, encrypting same, and forwarding the same into a Byte Stream Transmission stage;

e. A Byte Stream Transmission stage for conveying the framed, and optionally encrypted data packets into a modulator/demodulator section for transmission; and

f. A Transmission Management stage for assigning transmission timing slots for framed and optionally encrypted data that is conveyed into the modulator/demodulator section;

7. A system according to claim 6 wherein the Timer and Transmission Management stage assigns transmission timing slots based on information collected from the analysis of MAC frames by the Advanced

Packet Analysis stage of the receiving section, and on information received from the Flow Reception From Host stage of the transmission section;

8. A system according to claim 7, wherein the information for assigning transmission timing slots is stored in a memory that is shared by the transmitting section and the receiving section.

9. A system according to claim 4 further comprising a CMMS high level language residing in the Host, and a driver for interfacing between the CMMS and the microcode.

10. A system according to claim 4 further comprising a GUI high level language code for enabling a user of the process to monitor and configure the process.

11. A system according to claim 2 wherein the validity check is at least a checksum check.

12. A system according to claim 2 wherein the validity check is a checksum check and error correction.

13. A process according to claim 1 wherein the transmitting pipeline and the receiving pipeline are one way pipelines comprising plurality of segments.

14. A process according to claim 13 wherein the transmitting and the receiving pipelines comprise of hardware segments, software segments, or a combination of hardware and software segments.

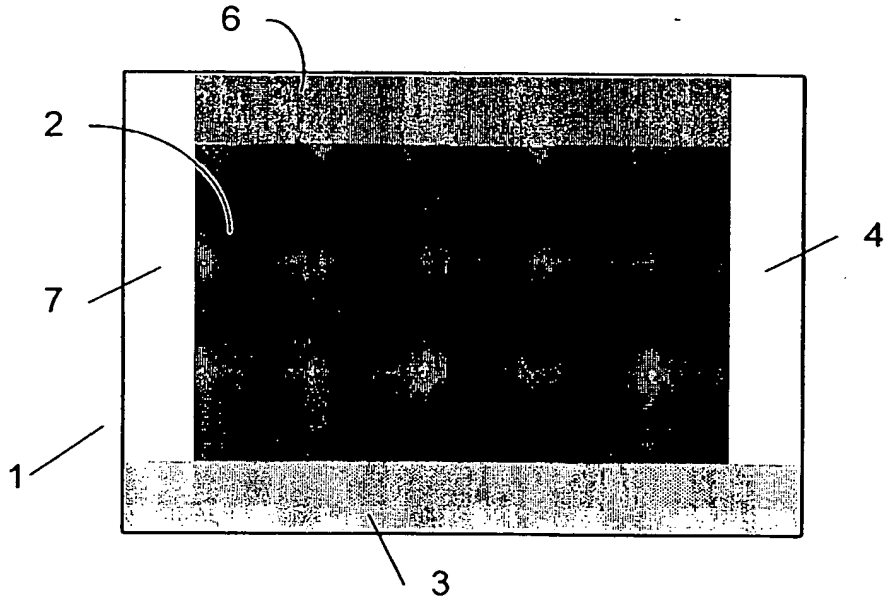

Fig. 1

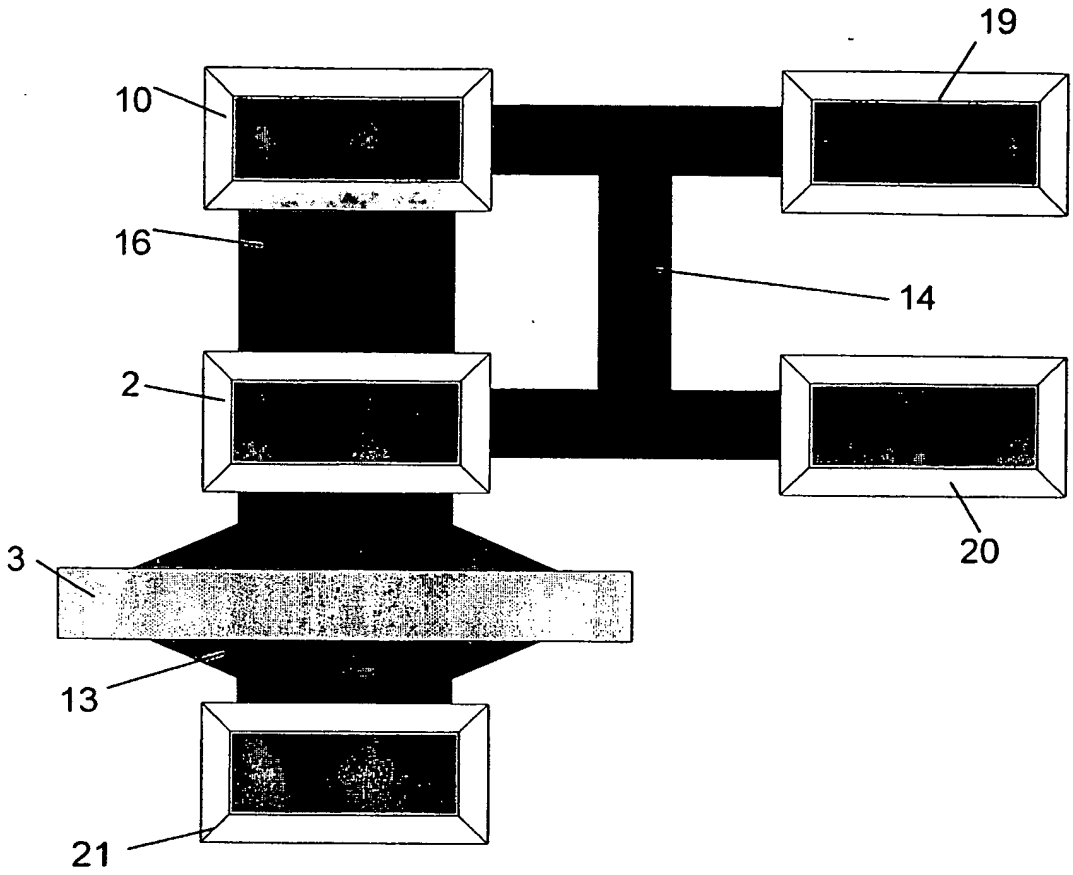

Fig. 2

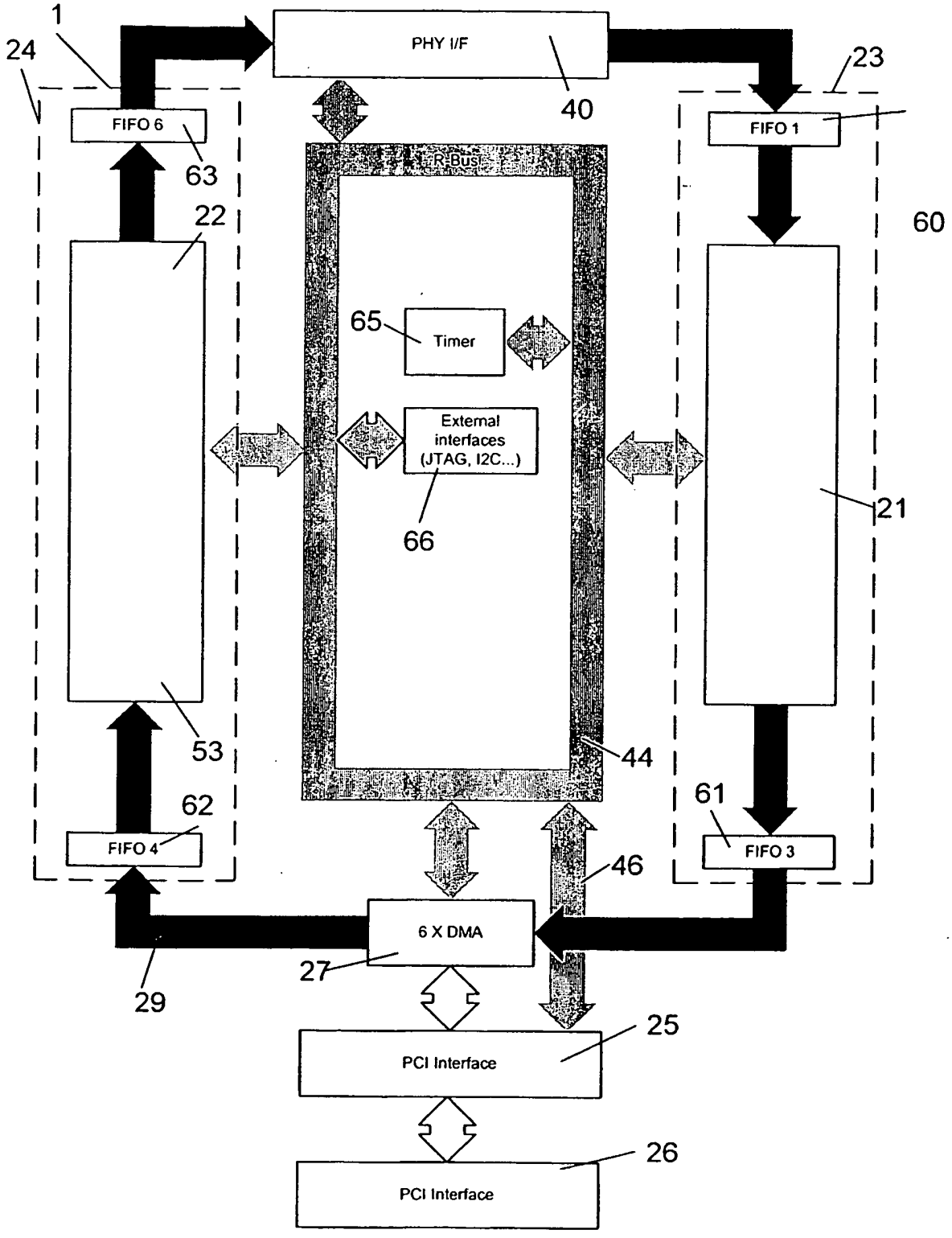
1
24
PHY I/F
40
23
FIFO 6
63
FIFO 1
R-Bus
60
22
65
Timer
External interfaces (JTAG, I2C...)
66
21
53
44
62
61
FIFO 4
FIFO 3
46
6 X DMA
27
29
25
PCI Interface
26
PCI Interface

Fig. 3

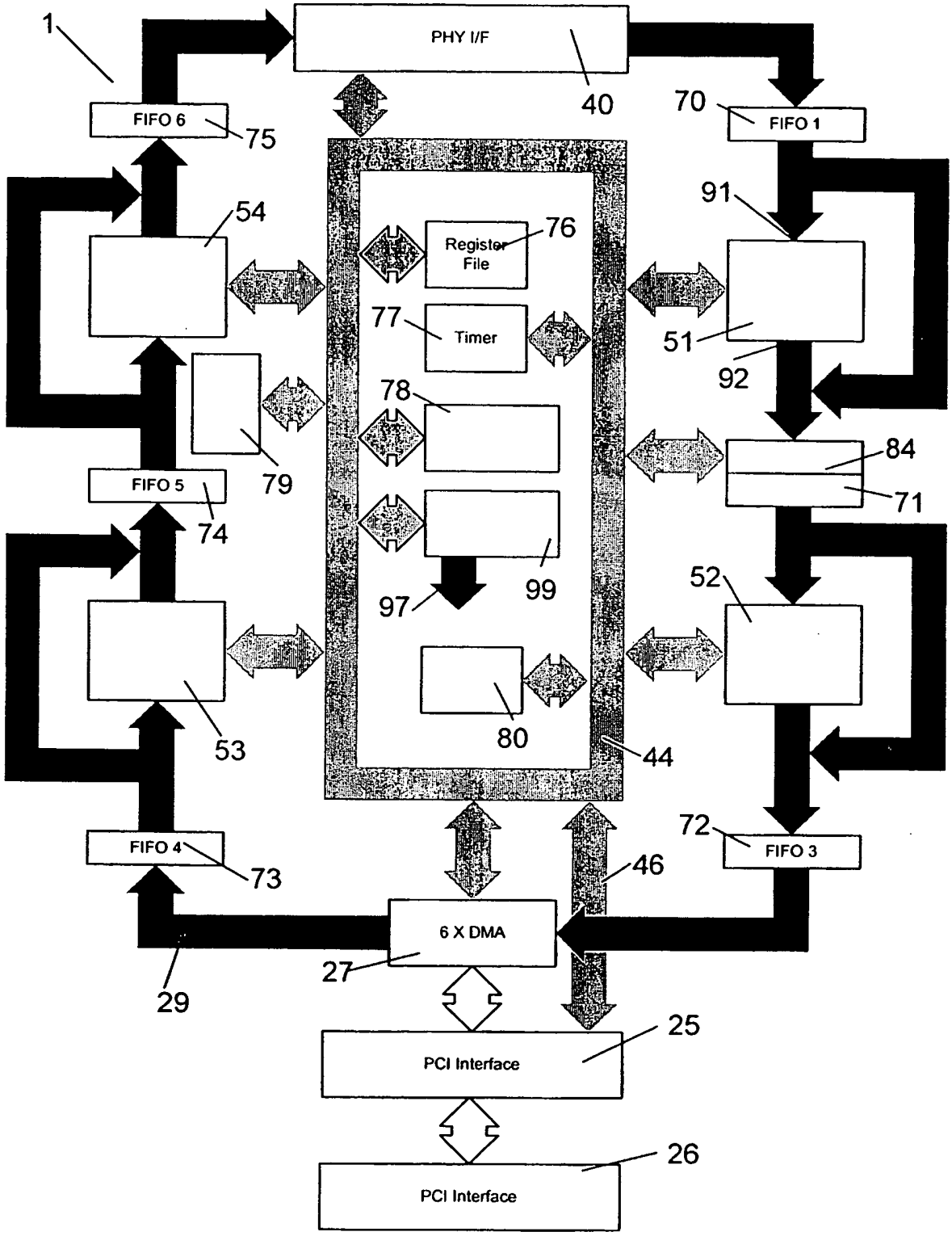
1
PHY I/F
40
70
FIFO 1
FIFO 6
75
91
54
Register File
76
77
Timer
51
92
78
84
79
71
FIFO 5
74
52
97
99
53
80
44
72
FIFO 4
73
FIFO 3
46
6 X DMA
27
29
25
PCI Interface
26
PCI Interface

Fig. 4

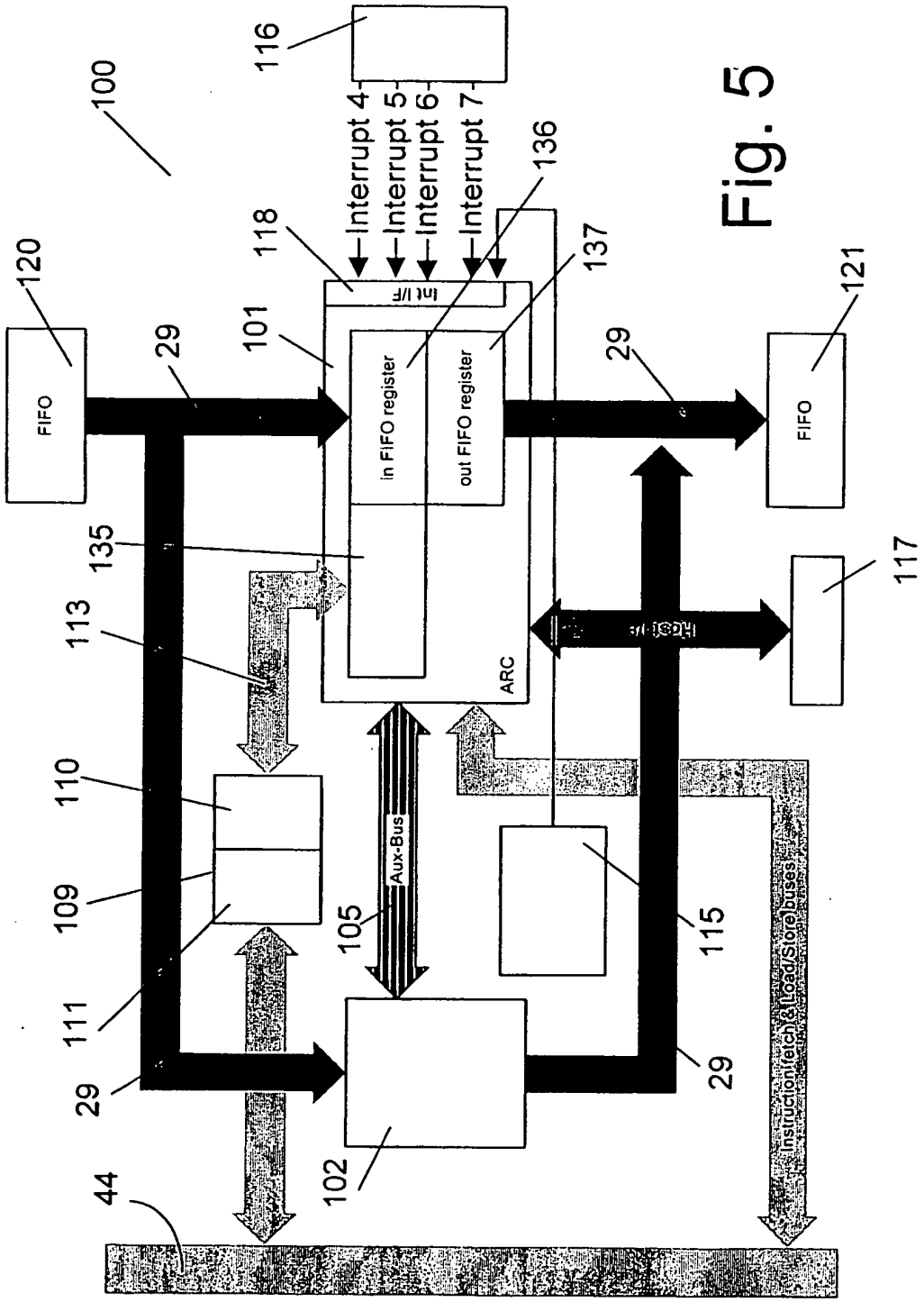
100
116
Interrupt 4
Interrupt 5
Interrupt 6
Interrupt 7
136
118
137
120
121
29
101
FIFO
in FIFO register
out FIFO register
FIFO
135
117
113
ARC
110
Aux-Bus
109
105
115
111
29
29
102
44
Instruction(fetch & Load/Store)buses

Fig. 5

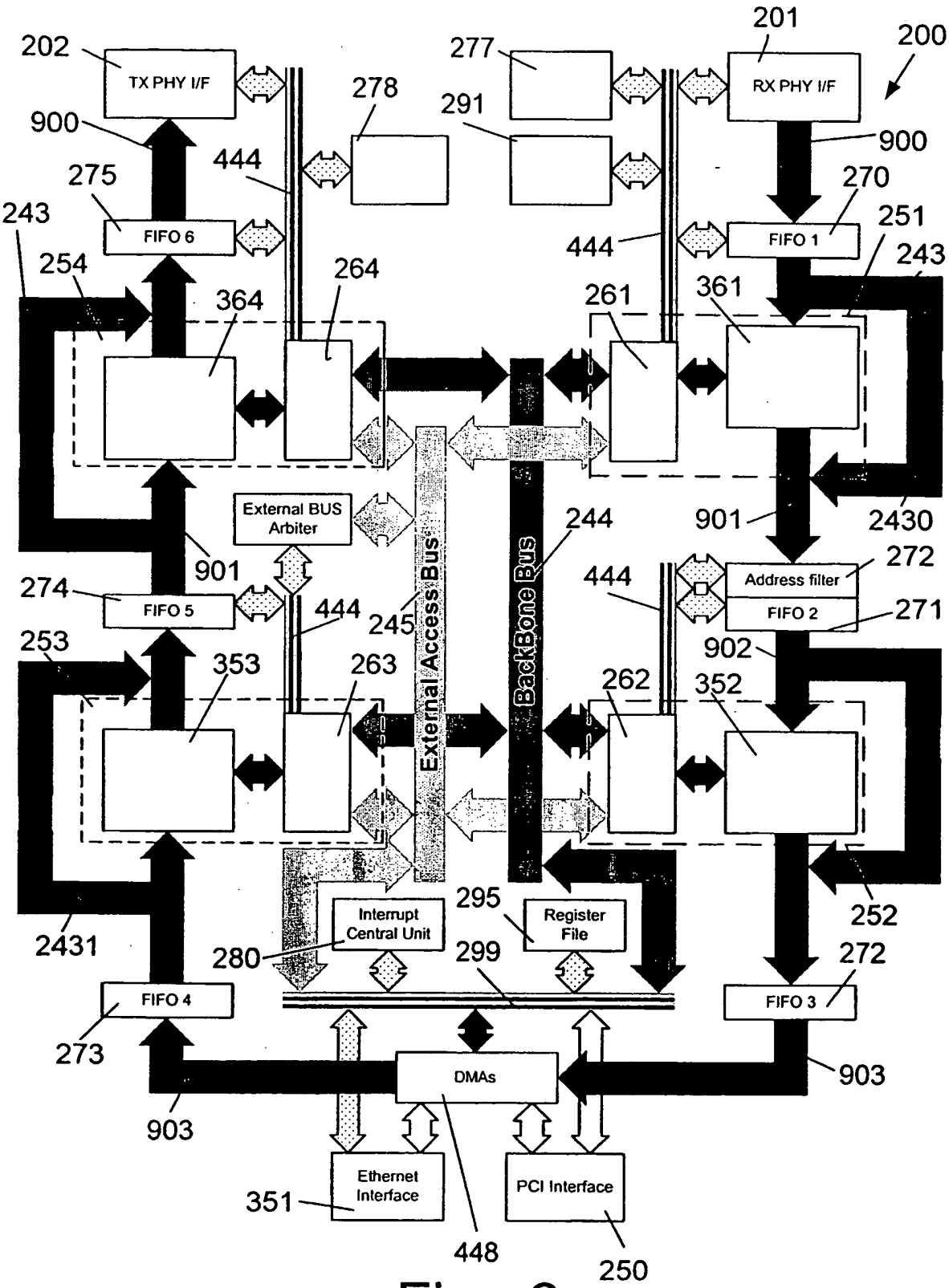
202
TX PHY I/F
277
201
RX PHY I/F
200
278
291
900
900
444
275
270
243
FIFO 6
444
FIFO 1
251
254
264
261
361
243
364
External BUS Arbiter
External Access Bus
BackBone Bus
244
901
2430
901
272
274
FIFO 5
444
245
444
Address filter
FIFO 2
271
253
902
353
263
262
352
Interrupt Central Unit
295
Register File
252
2431
280
299
272
FIFO 4
FIFO 3
273
DMAs
903
903
Ethernet Interface
PCI Interface
351
448
250

Fig. 6

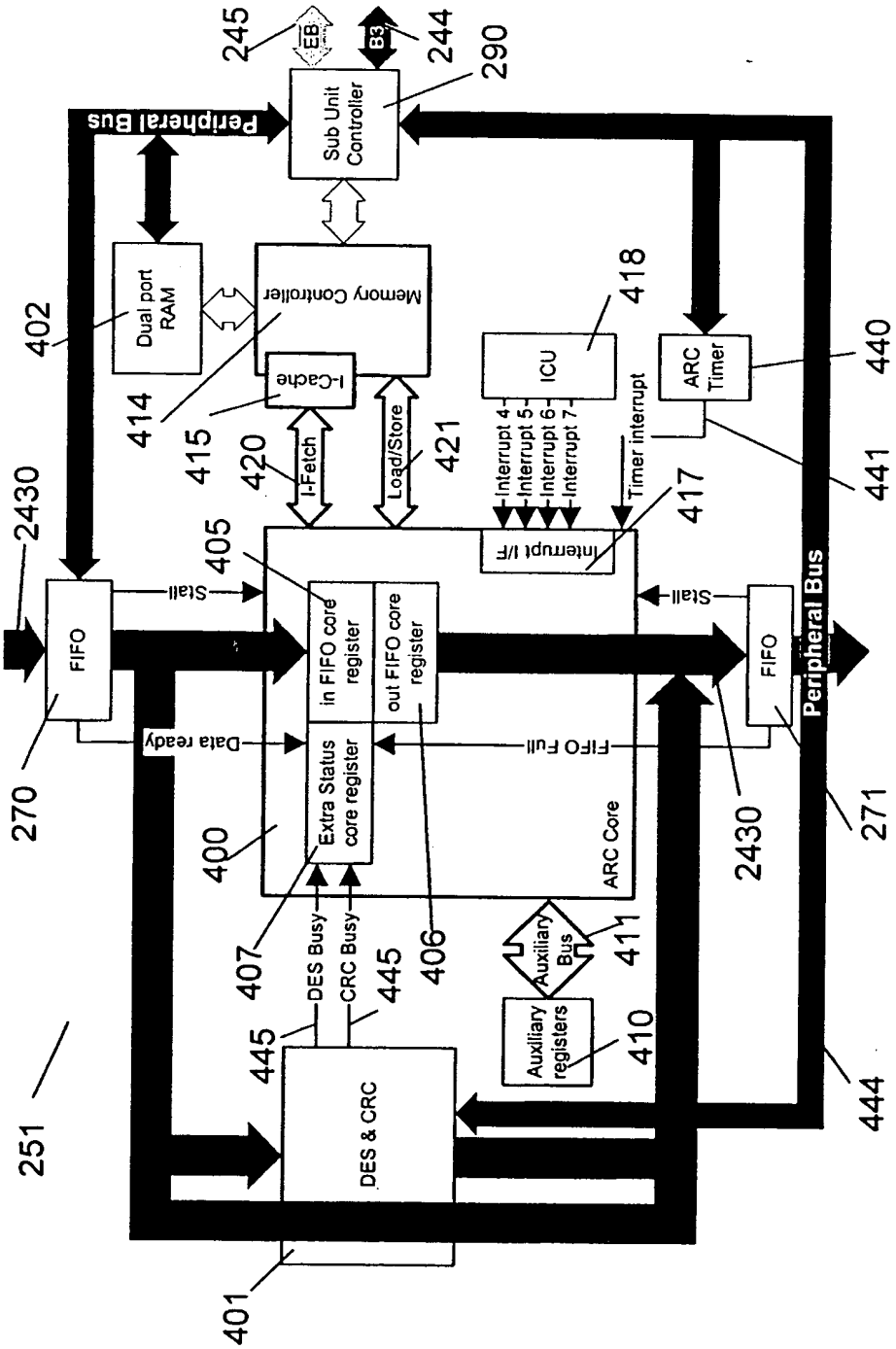
245
EB
B3
244
290
Peripheral Bus
Sub Unit Controller
Dual port RAM
Memory Controller
I-Cache
402
414
415
420
I-Fetch
Load/Store
421
ICU
418
ARC Timer
440
Interrupt 4
Interrupt 5
Interrupt 6
Interrupt 7
Timer interrupt
441
417
Interrupt I/F
2430
405
FIFO
Stall
in FIFO core register
out FIFO core register
Stall
FIFO
Peripheral Bus
Data ready
FIFO Full
270
Extra Status core register
400
ARC Core
2430
271
DES Busy
CRC Busy
407
406
445
Auxiliary Bus
411
Auxiliary registers
410
445
DES & CRC
444
251
401

Fig. 7

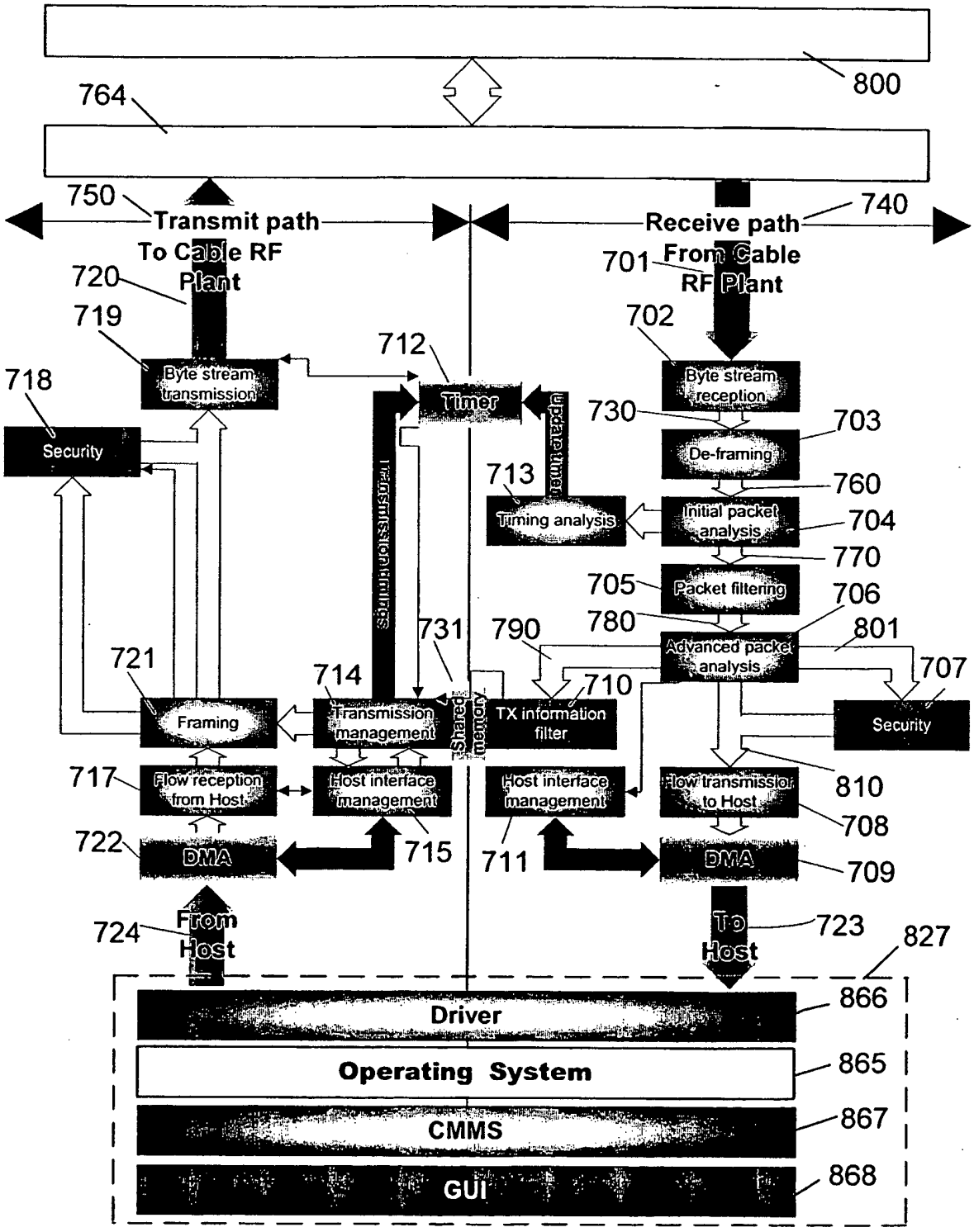
800
764
750
Transmit path
To Cable RF
Plant
740
Receive path
From Cable
RF Plant
701
720
719
712
702
718
Byte stream
transmission
Timer
Update timer
Byte stream
reception
730
703
Security
De-framing
713
760
Timing analysis
Initial packet
analysis
704
770
705
Packet filtering
706
780
790
731
801
Transmission timings
721
714
Advanced packet
analysis
707
710
Framing
Transmission
management
Shared memory
TX information
filter
Security
717
Flow reception
from Host
Host interface
management
Host interface
management
Flow transmission
to Host
810
722
DMA
715
711
DMA
708
709
724
From
Host
To
Host
723
827
Driver
866
Operating System
865
CMMS
867
GUI
868

Fig. 8

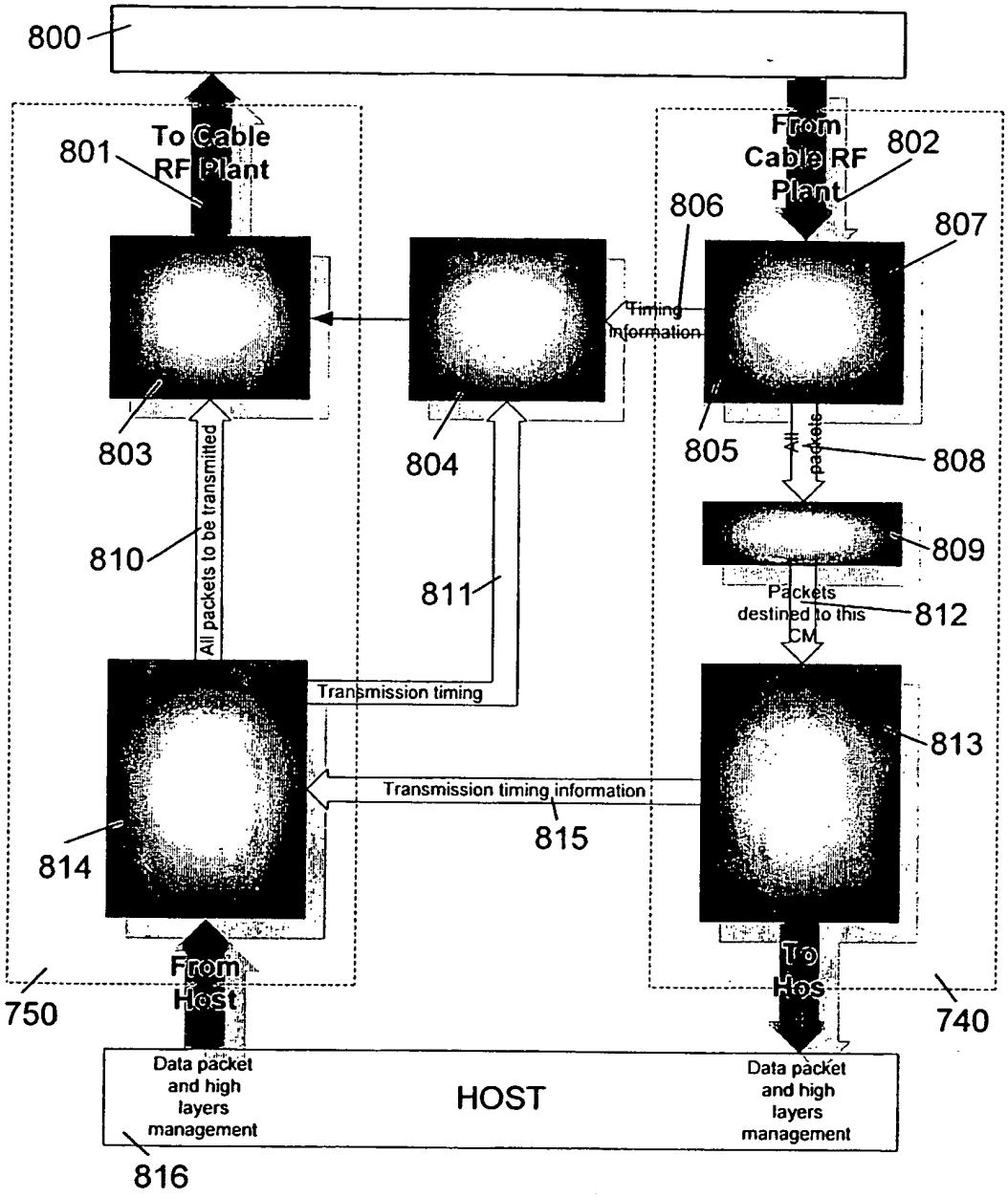
800
To Cable RF Plant
801
From Cable RF Plant
802
806
807
Timing information
803
804
805
All packets
808
809
All packets to be transmitted
810
811
Packets destined to this CM
812
Transmission timing
813
Transmission timing information
815
814
From Host
To Hos
750
740
Data packet and high layers management
HOST
Data packet and high layers management
816

Fig. 9

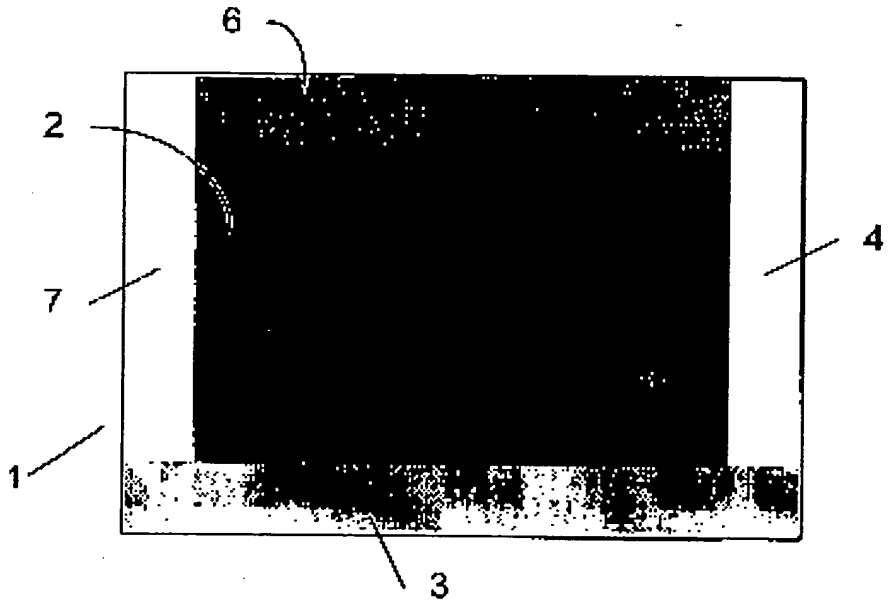

Fig. 1

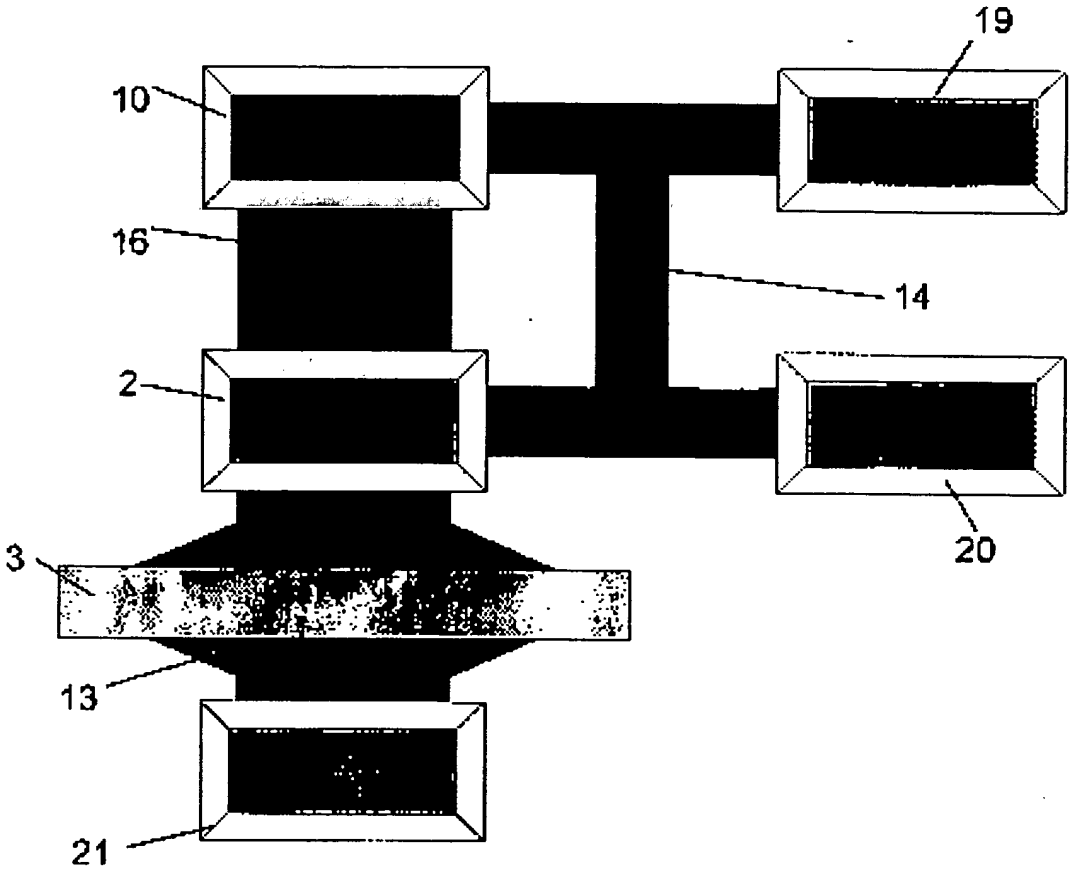

Fig. 2

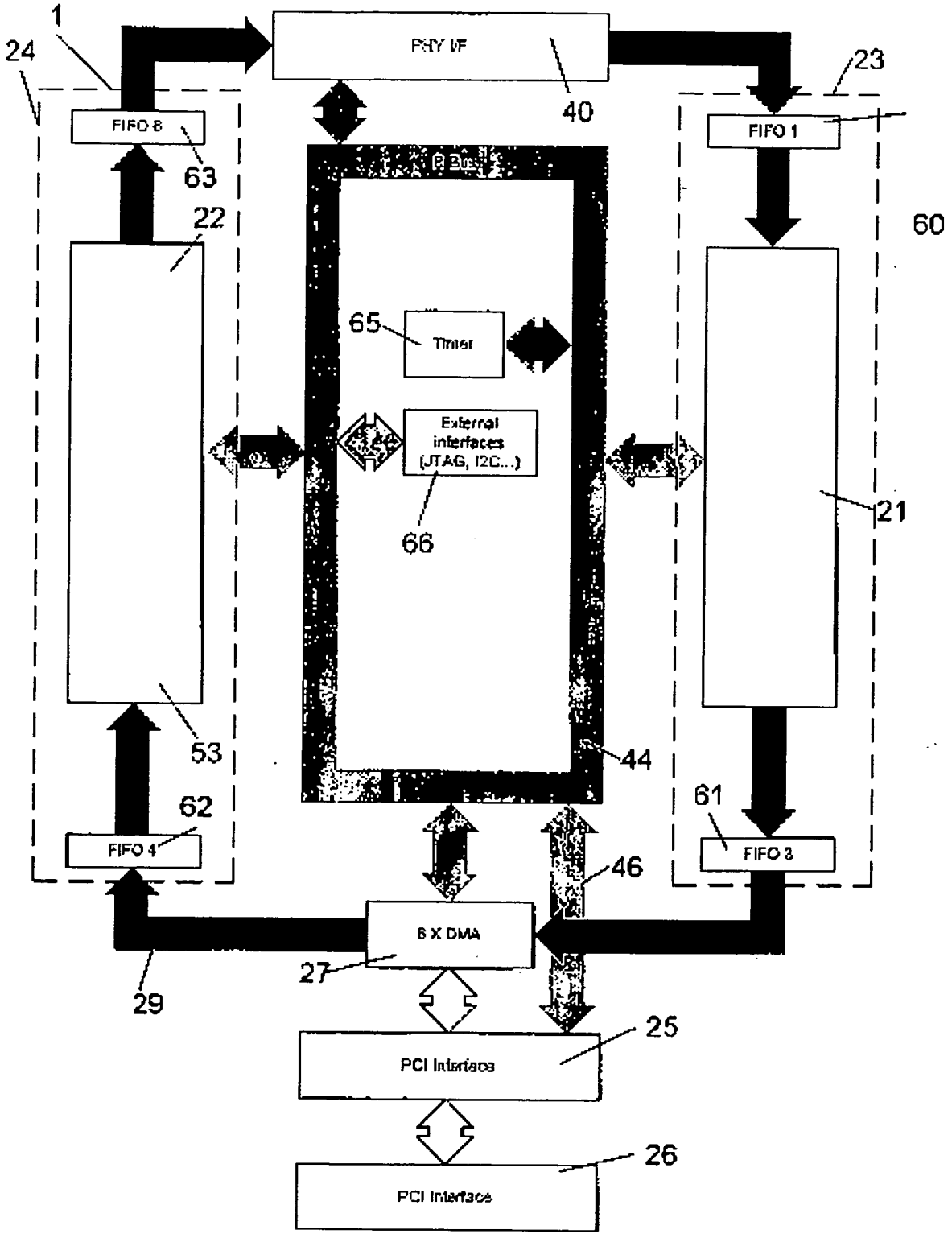
1
24
PHY I/F
40
23
FIFO 8
63
FIFO 1
22
60
65
Timer
External interfaces (JTAG, I2C...)
66
21
53
44
62
61
FIFO 4
FIFO 3
46
8 X DMA
27
29
25
PCI Interface
26
PCI Interface

Fig. 3

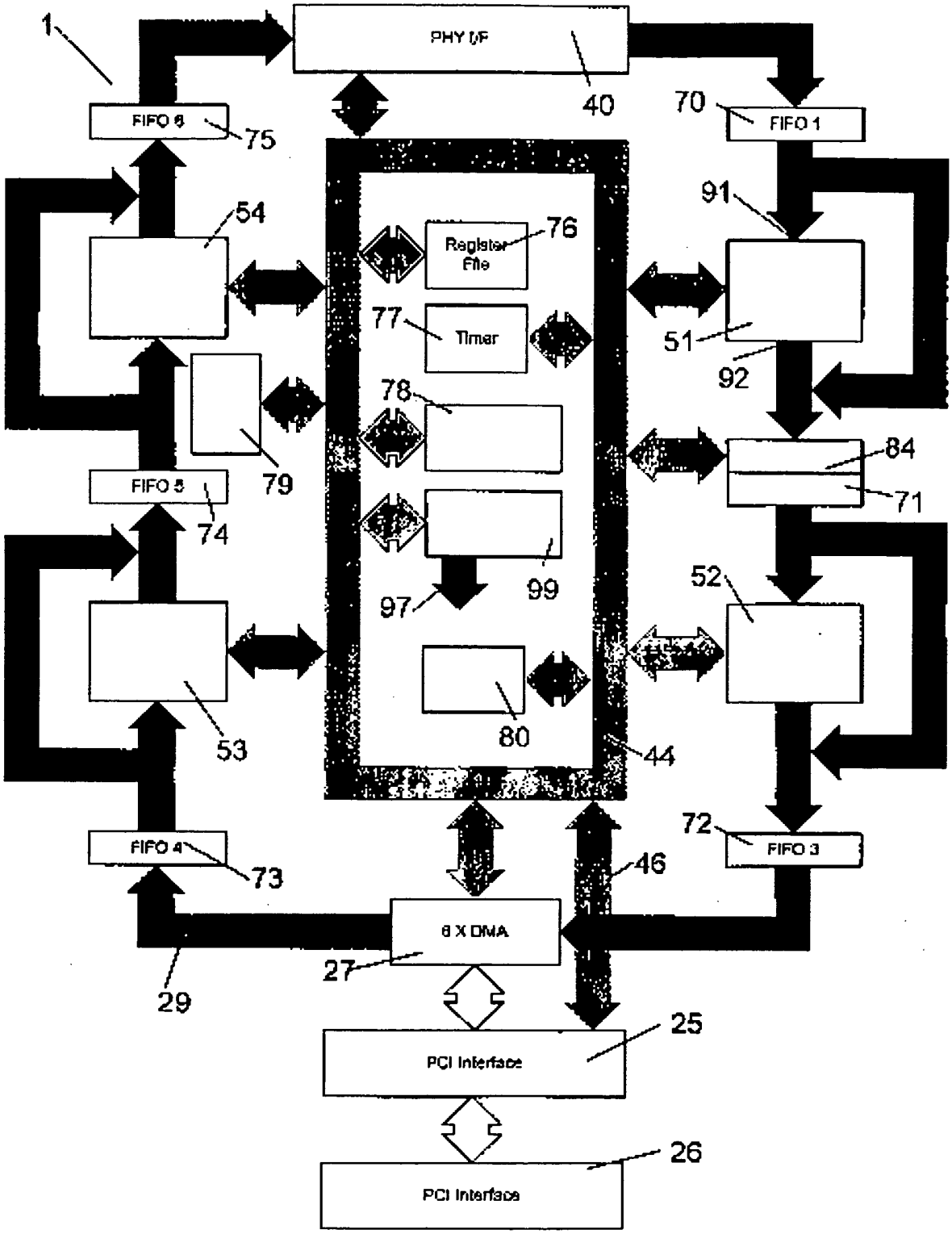
1
PHY I/F
40
70
FIFO 1
FIFO 6
75
54
Register File
76
77
Timer
91
51
92
78
79
84
71
FIFO 5
74
97
99
52
53
80
44
72
FIFO 4
73
FIFO 3
46
8 X DMA
27
29
25
PCI Interface
26
PCI Interface

Fig. 4

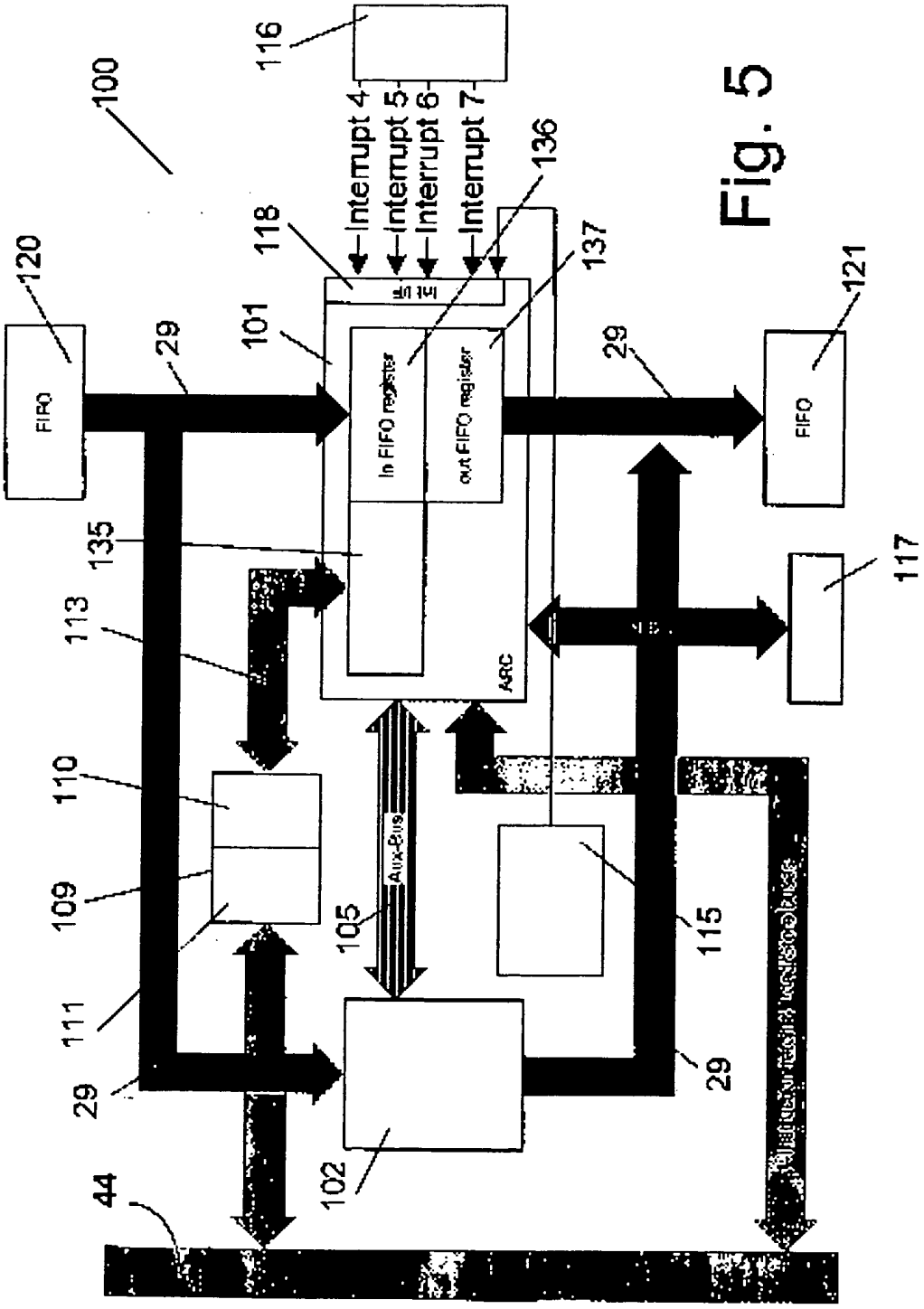

Fig. 5

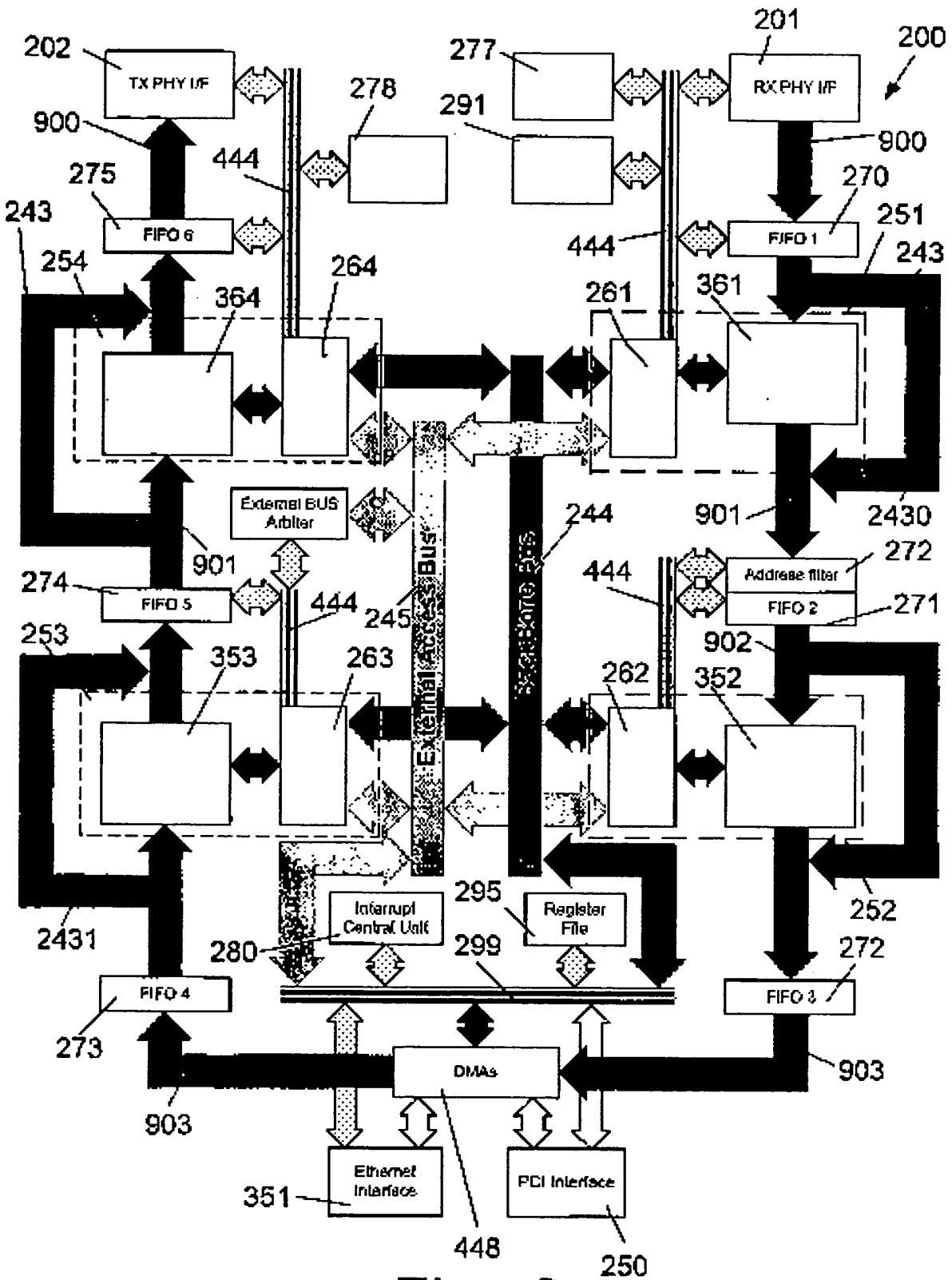

Fig. 6

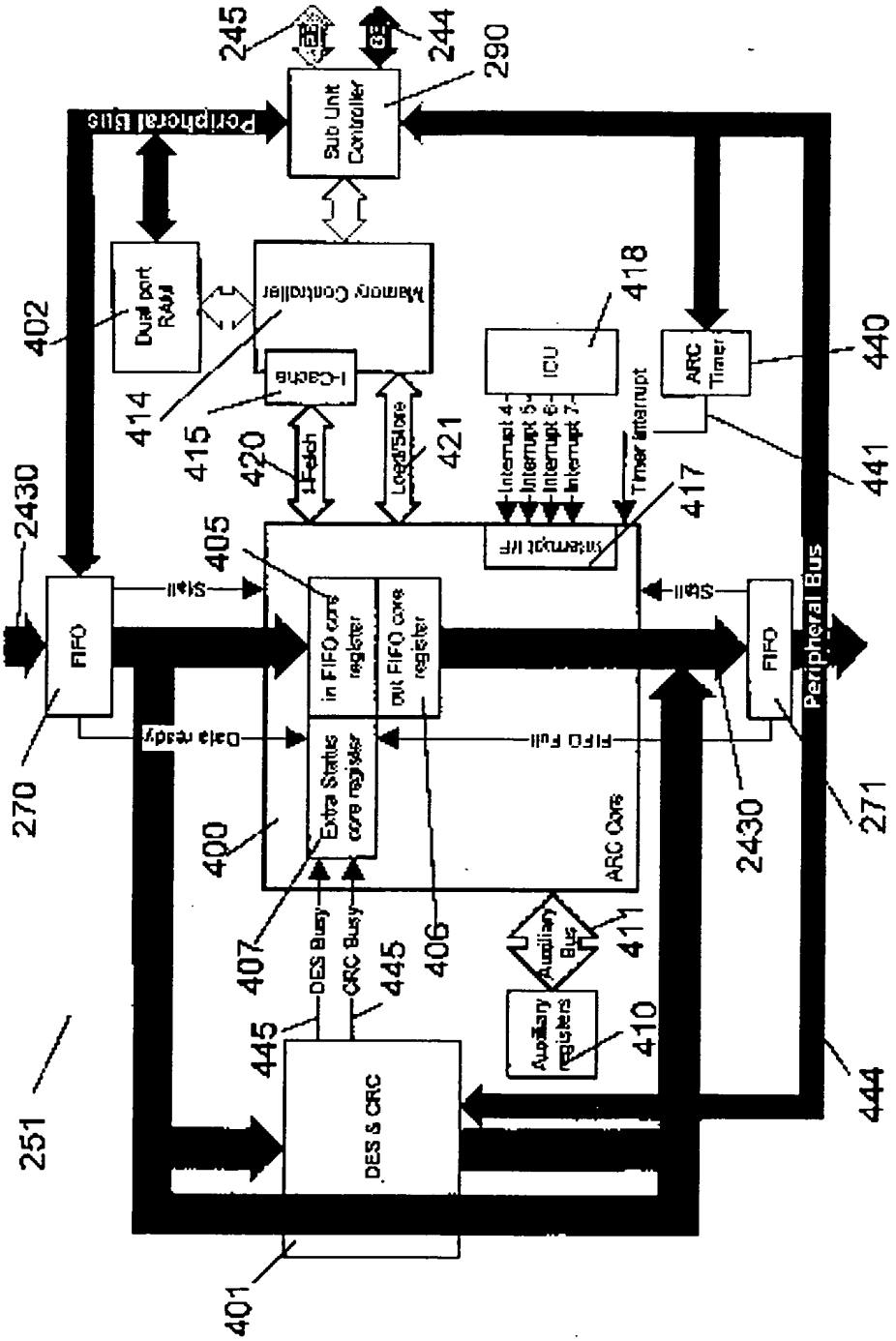
245
244
290
Peripheral Bus
Sub Unit Controller
Dual port RAM
Memory Controller
I-Cache
402
414
415
420
421
418
ICU
ARC Timer
440
441
Interrupt 4
Interrupt 5
Interrupt 6
Interrupt 7
Timer interrupt
417
Interrupt I/F
2430
405
FIFO
Slave
in FIFO core register
out FIFO core register
Data ready
FIFO Full
Extra Status core register
ARC Core
270
400
271
407
406
411
Auxiliary Bus
DES Busy
CRC Busy
445
Auxiliary registers
410
DES & CRC
444
251
401

Fig. 7

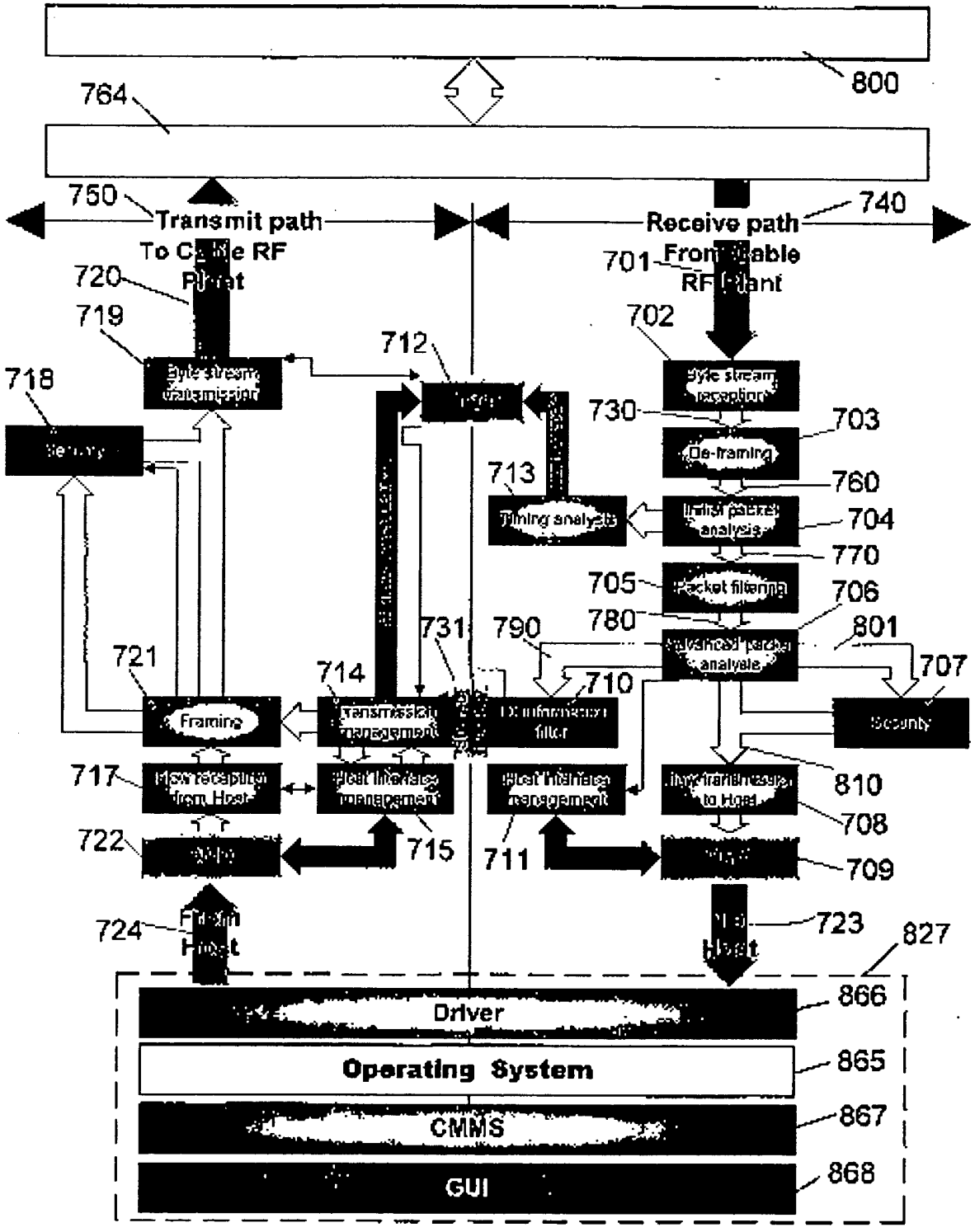
800
764
750
Transmit path
740
Receive path
720
719
712
718
730
703
713
760
704
770
705
706
731
790
780
801
721
714
710
707
717
715
711
810
708
722
709
724
723
827
866
Driver
865
Operating System
867
CMMS
868
GUI

Fig. 8

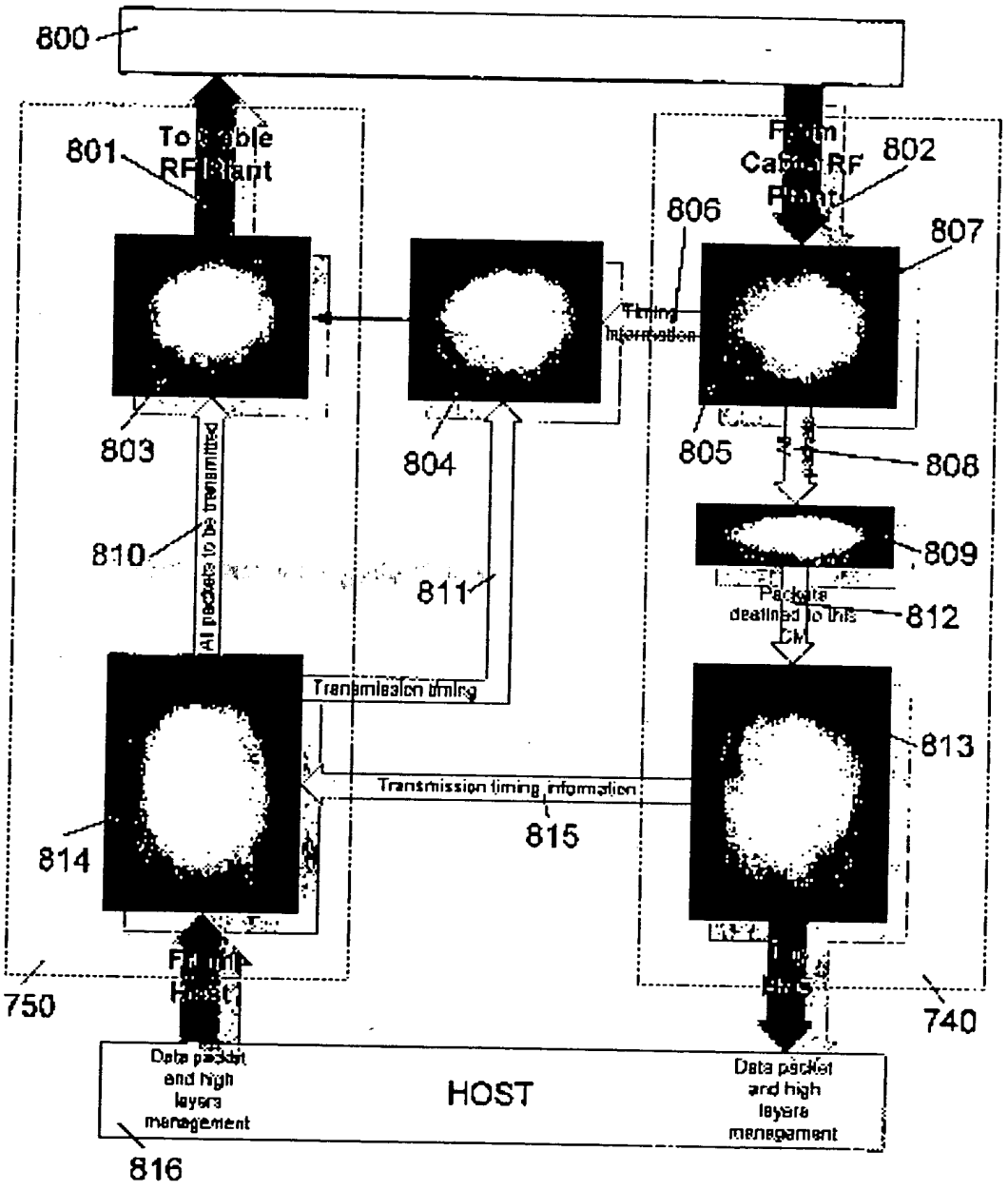
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
750
740
All packets to be transmitted
Transmission timing
Transmission timing information
Packets destined to this CM
HOST
Data packet and high layers management
Data packet and high layers management

Fig. 9